The Workshops of the Thirty-Second
AAAI Conference on Artificial Intelligence

State Tracking Networks for Dialog State Tracking

Xuguang Wang, Xingyi Cheng,* Jie Zhou, Wei Xu
Baidu Research - Institue of Deep Learning
{chengxingyi,zhangruiqing01,zhoujie01,wei.xu}@baidu.com

Abstract

Dialog state tracking is to accurately infer a compact representation of the dialog status up to the current turn, it needs to summarize all the dialog history information and user's goals. In a successful spoken dialog system, dialog state tracker is one of the most important components of the pipelines. Yet until recently, there are no general, flexible, accurate and truly end to end dialog state tracking models. In this paper, we propose a novel model named state tracking networks that can perform dialog state tracking in a natural efficient and elegant way. It uses an explicit gate to model the state updating mechanism and can be trained end to end in a deterministic manner using standard backpropagation techniques or stochastically by reinforcement learning. Our model can both deal with ASR and text input without any modification. We perform experiments on the Second Dialog State Tracking Challenge dataset(DSTC2) and get performance matching the state-of-the-art models. Furthermore, the qualitative analysis reveals that the gating mechanism learned by our model agree well with intuition.

1 Introduction

Conversational agent or spoken dialog system become more and more popular nowadays due to great progress of audio and natural language processing techniques(Vinyals and Le 2015; Shang, Lu, and Li 2015; Wen et al. 2016; Bordes and Weston 2016), it is the most natural way that humans interact with the digital devices. When a conversation is going on, one of the most important thing for spoken dialog systems is to keep tracking the state of this conversation. The state of the conversation summarize all the necessary dialog history information and is critical for the dialog policy model which decide the action the system should perform next(Williams, Raux, and Henderson 2016).

The Dialog State Tracking Challenge(DSTC)(Henderson, Thomson, and Williams 2014) series of tasks introduced the shared testbed and evaluation metrics for dialog state tracking. The DSTC2 dataset is about restaurant reservation scenario where the dialog system interact with the user(through speech recognizer) who want to book a restaurant. After each user utterance, the system must infer the state of the dialog which contain the user-specified constraints(inform) and so on[1]. We take a example dialog from the DSTC2 training set[2] to show the ground truth of dialog state tracking(Table 1):

When confronted with noisy speech recognition, varied natural language representation and long dependency of dialog history, dialog state tracking become a non-trivial task. While traditional methods firstly using spoken language understanding(SLU)(Williams and Young 2007) to convert user utterance to meaning representation and then pass this

*Xingyi Cheng is the corresponding author.
Copyright © 2018, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

[1]Actually, in DSTC2, dialog state contains three components: Goals, Requested slots, and Method, the Goals corresponding to constraints(inform), the Requested slots refer to the user's request for slot value of a specified restaurant, the 'method' describes the way the user is trying to interact with the system. The last two components can be determined by the current system and user turn without the need of dialog history, so we omit it from our work.

[2]In DSTC2, user utterance is represent as automatic speech recognition(ASR) top-n results.

Table 1: A example dialog from the DSTC2 training set with ground truth dialog state(Requested slots and Method are omitted here).

Turn	Utterance, State
1	**system**: Hello, welcome to the Cambridge restaurant system? You can ask for restaurants by area, price range or food type. How may I help you? **user**: im looking for an expensive restaurant that serves venetian food **State**: inform(food=venetian, price=expensive)
2	**system**: Sorry there is no venetian restaurant in the expensive price range **user**: turkish food **State**: inform(food=turkish, price=expensive)
3	**system**: meze bar restaurant is a great restaurant serving turkish food and it is in the expensive price range **user**: can i get the addresses **State**: inform(food=turkish, price=expensive)
4	**system**: Sure , meze bar restaurant is on 196 Mill Road City Centre **user**: thank you good bye **State**: inform(food=turkish, price=expensive)

SLU result to the dialog state tracker(DST), recently arising neural networks based methods bypass the SLU procedure and require little feature engineering(Perez and Liu 2016; Mrksic et al. 2017; Henderson, Thomson, and Young 2014b). However, these neural methods tend to share two common limitations. First, they often rely on the meaning representation of system utterance, when training with human-human dialog data, they must first process the system part utterance to its meaning representation. Second, they often train each individual tracker(e.g. individual parameters) for each slot(e.g. food, price range, area) and don't leverage the correlation between those slots information.

In this paper, we propose a new model named state tracking networks(STN) for an end to end dialog state tracking. Our model receives original system and user utterance(ASR or text) input, take history information into consideration and process all of the slots jointly. STN use an explicit gate to model the state updating mechanism with interpretability. We evaluate STN on DSTC2 dataset and get performance matching the state-of-the-art models that are more computational sophisticated than ours.

2 State Tracking Networks

Our state tracking networks are designed to track the dialog state end to end and consists of three components: an convolutional input encoder that get distributed representation of system and user utterance and their interaction, a state updater which update the distributed representation of dialog state and state output module which output the probability form of dialog state. Before we describe our model, we briefly introduce the procedure of dialog state tracking using our model[3].

In each turn of the dialog[4], our model first extract useful information from system utterance and user utterance, the interaction of theirs, then, based on the information we extract, we compute gate for each slot that determines whether the slot value should be updated or not, for the slot need to be updated, we use the computed distributed representation of candidate slot values, for the one shouldn't be updated, we keep the last turn's value unchanged. At last, we cast the distributed representation of slot values of each slot to their probabilistic forms. This procedure corresponding to the three part of our state tracking networks which will be described in detail next.

2.1 Convolutional Input Encoder

Convolutional neural network(CNN) is one of the most powerful models for modeling sentence(Collobert et al. 2011; Kim 2014). Our input encoder is built upon convolutional neural networks. Let $\mathbf{x}_i \in \mathbb{R}^k$ be the k-dimensional word embedding corresponding to the i-th word in the sentence. A n-gram concatenated vector(start from word position i) can be represented as:

$$\mathbf{x}_{i:i+n-1} = [\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+n-1}] \tag{1}$$

Where $[\cdot, \cdot, \cdot]$ denote vector concatenation. Processing each of these n-gram concatenated vectors through convolution operations we get a set of feature maps $\{\mathbf{h}_{n,i}\}$, where:

$$\mathbf{h}_{n,i} = f(\mathbf{W}_n \mathbf{x}_{i:i+n-1}^T + \mathbf{b}_n) \tag{2}$$

f is the non-linearity activation function. Then, we apply max-pooling over the features map of the convolution size n and denote it as:

$$\mathbf{c}_n = \max_i \{\mathbf{h}_{n,i}\} \tag{3}$$

The max-pooling over these vectors is a coordinate-wise max operation. For different window sizes $n_1, n_2, \ldots$ of the processed vector, we concatenate them to form the whole sentence representation vector:

$$\mathbf{c} = [\mathbf{c}_{n1}, \mathbf{c}_{n2}, \ldots] \tag{4}$$

Because the original user utterance is represented as ASR top-n list, which is a list of pairs of sentence and its probability:$\{(\mathbf{s}_1, p_1), (\mathbf{s}_2, p_2), \ldots, (\mathbf{s}_n, p_n)\}$. We extend our convolutional sentence representation to the case of ASR top-n list. The method is quite simple: we build all of the n-grams in the ASR top-n list, computer probability for each n-gram and take it as the weight for the n-gram, then, replace Eq. (1) with the weighted concatenated vector: $\mathbf{x}_{i:i+n-1} = p_{n,i}[\mathbf{x}_i, \mathbf{x}_{i+1}, \ldots, \mathbf{x}_{i+n-1}]$ and keep the other procedure unchanged. This way, the text form utterance can be viewed as the special case where all n-gram weight is equal to one. We use different parametric CNN to process the system utterance and user utterance and denote them as $\mathbf{s}, \mathbf{u}$ respectively. For the interaction of the system and user utterance, we simply feed $[\mathbf{s}, \mathbf{u}]$ to a feedforward neural network(FNN) with one hidden and output layer. At last, we use $[\mathbf{s}, \mathbf{u}, \mathrm{FNN}([\mathbf{s}, \mathbf{u}])]$ to denote the turn representation. For simplicity, at turn t, we use column vector $\mathbf{i}_t$ to represent the encoded turn information:

$$\mathbf{i}_t = [\mathbf{s}_t, \mathbf{u}_t, \mathrm{FNN}([\mathbf{s}_t, \mathbf{u}_t])]^T \tag{5}$$

2.2 State Updater

For state updater, we first compute gate for each slot which determines how much this slot should be updated:

$$\mathbf{g}_t^j = \mathrm{sigmoid}(\mathbf{i}_t^T \mathbf{A} \mathbf{s}^j) \tag{6}$$

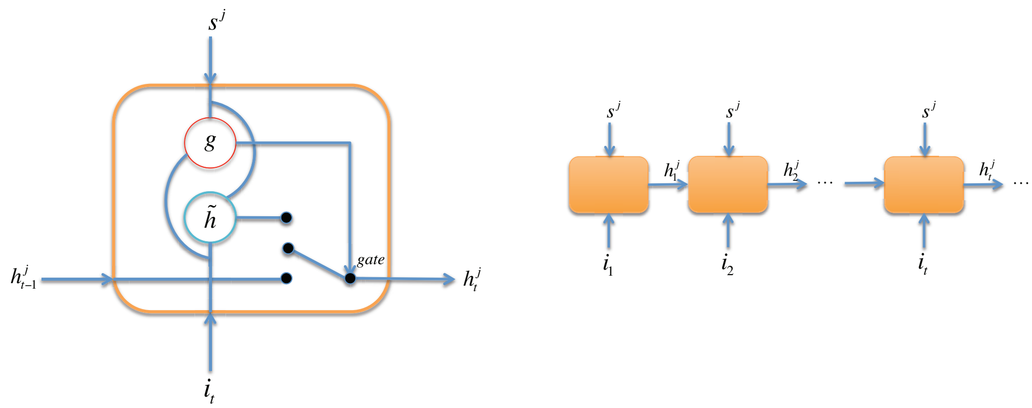
where $\mathbf{g}_t^j$ is the slot j's gate at turning t and it is a scalar[5]. $\mathbf{s}^j$ is the slot j's embedding parameter that needs to be learned. $\mathbf{A}$ is parameter matrix. Then, we compute the distributed representation of candidate slot value which is used for the update:

$$\widetilde{\mathbf{h}}_t^j = \tanh(\mathbf{V}\mathbf{i}_t + \mathbf{W}\mathbf{s}^j + \mathbf{b}) \tag{7}$$

[3]Because Requested slots and Method of the dialog state doesn't require dialog history information, we only care about the constraints(inform) part of the dialog state, namely the slot-value pairs(e.g. food=venetian, pricerange=expensive).

[4]Each turn contain a system turn and an user turn

[5]In experiment, we use activation function hard_sigmoid instead of sigmoid in order to get the extreme value of 0 or 1. In all the following context, we use hard_sigmoid also.



(a) (b)

Figure 1: (a) State update at turn t for slot j: the gate $\mathbf{g}$ take slot embedding $\mathbf{s}^j$ and turn embedding $\mathbf{i}_t$ as input and determine how much the state should be updated, $\widetilde{\mathbf{h}}$ take the same input and output the candidate state which is used for update, the ultimate state at this turn is a combination of $\widetilde{\mathbf{h}}$ and last state $\mathbf{h}_{t-1}^j$. (b) Overview of state update network unfolded on turn dimension.

where $\mathbf{V}, \mathbf{W}, \mathbf{b}$ are learnable parameters. The update formula is:

$$\mathbf{h}_t^j = (1 - \mathbf{g}_t^j)\mathbf{h}_{t-1}^j + \mathbf{g}_t^j \widetilde{\mathbf{h}}_t^j \tag{8}$$

which is a combination of last time state and current turn candidate state, it is controlled by the gate we have computed, as show in Fig. 1. Until now, the gate is a continues value lie in $(0, 1)$. It is the "soft" form of gate and we can design "hard" version by append Eq. (6) with a further round operation:

$$\mathbf{g}_t^j = \text{round}(\text{sigmoid}(\mathbf{i}_t^T \mathbf{A}\mathbf{s}^j)) \tag{9}$$

the hard version gate can only be 0 or 1. For value in $(0, 1)$, if greater than 0.5, it round to 1, else 0. Eq. (9) can not be used for training and we will introduce its training form in section3.

2.3 State Output

After we have got the distributed representation of the state $\mathbf{h}_t^j$, compute its probabilistic distribution is straightforward, for slot j at turn t:

$$p_t^j = \text{sotfmax}(\mathbf{C}^j \mathbf{h}_t^j + \mathbf{b}^j) \tag{10}$$

Where p_t^j denote the probability of each value[6] for slot j, its dimension is equal to the available values nums plus two[7]. $\mathbf{C}^j$ and $\mathbf{b}^j$ are slot j's parameter, they differ from slot to slot. Joint p_t^j for each j, we get the state probability at turn t.

[6]We observe that the current turn's slot value most probably be the one which exists in the dialog history(including this turn), so, when predicting, we mask the candidate values which don't appear in the history dialog.

[7]Two special values: 'dontcare' and 'not mentioned'.

3 Training

When we use the soft gate(6), the state tracking networks can be differentiable end to end, the loss is negative log-likelihood:

$$\mathcal{L}_{network} = -\sum_{t,j} \log p_t^j(v_t^j) \tag{11}$$

Where, v_t^j is the target value of slot j at turn t, $p_t^j(v_t^j)$ is its probability output by our state tracking networks. However, when we use the hard gate, this part is't differentiable. When training, we replace Eq. (9) with the stochastic version:

$$\begin{aligned} \mathbf{g}_t^j &= \text{bernoulli}(\text{sigmoid}(\mathbf{i}_t^T \mathbf{A}\mathbf{s}^j)) \\ &= \begin{cases} 1 & \text{with probability } \text{sigmoid}(\mathbf{i}_t^T \mathbf{A}\mathbf{s}^j), \\ 0 & \text{with probability } 1 - \text{sigmoid}(\mathbf{i}_t^T \mathbf{A}\mathbf{s}^j). \end{cases} \end{aligned} \tag{12}$$

Neural networks with stochastic unit(Schulman et al. 2015) can be optimized by maximizing an approximate variational lower bound or equivalently by RENINFORCE(Williams 1992), in this place we use the RENINFORCE algorithm and the gate loss can be written as:

$$\mathcal{L}_{gate} = -\sum_{t,j} \log p(\mathbf{g}_t^j | \mathbf{i}_t)(G_t^j - b_t^j) \tag{13}$$

Where G_t^j is the cumulative discounted log-likelihood of state tracking networks outputs influenced by $\mathbf{g}_t^j$:

$$G_t^j = \sum_{t'=t}^{L} \gamma^{t'-t} \log p_{t'}^j(v_{t'}^j) \tag{14}$$

Where, L is the total turn num of a dialog, γ is the discounting factor and we choose it to be 0.9. b_t^j is the baseline and $b_t^j = i_t^T A u^j$, u^j is the parameter whose size is equal to s^j. The lose of baseline is:

$$\mathcal{L}_{baseline} = \frac{1}{2} \sum_{t,j} (G_t^j - b_t^j)^2 \tag{15}$$

Table 2: Dialog state tracking result: slot prediction accuracy on DSTC2 test set

Type	Model	Area	Food	Price	Joint
ASR n-best	RNN(no dict.)(Henderson, Thomson, and Young 2014b; 2014a)	0.92	0.86	0.86	0.69
	RNN(sem. dict.)(Henderson, Thomson, and Young 2014b; 2014a)	0.91	0.86	0.93	0.73
	NBT-DNN(Mrksic et al. 2017)	0.90	0.84	0.94	0.72
	NBT-CNN(Mrksic et al. 2017)	0.90	0.83	0.93	0.72
	STN-soft	0.90	0.84	0.91	0.71
text	MemN2N(Perez and Liu 2016)	0.89	0.88	0.95	0.74
	STN-soft	0.95	0.93	0.95	0.84
	STN-hard	0.93	0.92	0.97	0.84
	STN-hard(soft init.)	0.95	0.94	0.92	0.84

At last, we sum up all of the loss and calculate the gradient[8]:

$$\begin{aligned}\nabla(\mathcal{L}_{total}) =& \nabla(\mathcal{L}_{network} + \mathcal{L}_{gate} + \mathcal{L}_{baseline})\\ =& -\sum_{t,j} \nabla \log p_t^j(v_t^j)\\ &- \sum_{t,j} \nabla \log p(g_t^j|i_t)(G_t^j - b_t^j)\\ &- \sum_{t,j} \nabla b_t^j(G_t^j - b_t^j)\end{aligned} \tag{16}$$

For training, we use the Adam(Kingma and Ba 2014) algorithm and choose the default learning rate 0.001, batch size is 50 dialogs. We apply dropout(Zaremba, Sutskever, and Vinyals 2014) to the output of convolutional encoder(equation(4)) and the input of state output model(section2.3, h_t^j of equation(10)) with dropout rate 0.2. We use the 50 dimensional Glove vectors(Pennington, Socher, and Manning 2014) as our word embeddings and keep them fixed during training. For convolutional encoder, window size of one and two grams are used and each with 100 filters. All the other hidden layer dimension is set to 200. We train our model for 100 pass, all the hype-parameters are determined by the performance of development set.

4 Experiments

4.1 DataSet

We experiment upon the DSTC2 dataset which come from the Second Dialog State Tracking Challenge(Henderson, Thomson, and Williams 2014). The dialog system interact with a user who want to find a specified restaurant around Cambridge, UK. The user can constrain the restaurant search by three informable slot: food type, area and price. As a dialog progress, the dialog system ask the user for slot information, the user answer these questions and can change the slot value of answered question. At the end of the dialog, the system suggests a restaurant, the user can ask for address, phone number and etc. of the restaurant. In this context, the dialog state tracker should track the value of each slot(food type, area, price) the user has specified at each turn.

In DSTC2, The user utterance has two kinds of recorded forms: ASR n-best list[9] and transcription text[10]. We perform experiment on both of them and report comparative experimental result. In all of our experiments, we directly take the system transcriptions as input instead of semantic form of system transcriptions which have been taken by others(Mrksic et al. 2017; Williams, Raux, and Henderson 2016; Henderson, Thomson, and Young 2014b; 2014a). The official transcriptions contain various spelling errors, we use the manually corrected version of Mrksic et al.. The training/development/test dataset contains 1612/506/1117 dialogs respectively.

4.2 Results

Table 2 presents dialog state tracking accuracy on DSTC2 test set of our state tracking networks(STN) and other comparative methods. On the ASR n-best list user utterance input type, we compare with two kinds of works: A delexicalisation-based RNN model(Henderson, Thomson, and Young 2014b; 2014a) with and without a hand-crafted semantic dictionary and the Neural belief tracker(NBT)(Mrksic et al. 2017) which is a data driven method but use a rule based state updater. We get comparative accuracy for all of the slots even though all the other models use semantic output of system transcriptions instead of the original utterance text. In this noisy input condition, we find the hard version STN perform a little worse than the soft version, so we only report the soft STN result. When the dialog utterance is noisy, the gate of state updater(8) is better described as uncertain probability form(soft) than certain deterministic form(hard), so the superior of soft version STN can be explained.

When experiment on the input type of user utterance transcriptions(text), we compare with the memory network approach(Perez and Liu 2016) which formulate the dialog state

[8]When we compute the gradient of $\mathcal{L}_{gate}$ (equation(13)), we take $(G_t^j - b_t^j)$ as constant, also, when compute the gradient of $\mathcal{L}_{baseline}$ (equation(15)), we take G_t^j as constant.

[9]Due to speech recognition errors, the ASR n-best list is very noisy.

[10]When use transcriptions, we actually tackle with the text dialog problems

Table 3: State and gate predicted by STN of a test sample in DSTC2: The state and gate predict by our soft STN and hard STN is all the same, hard gate can be the value of 0 or 1, soft gate can also reach the value of 0 or 1 due to the use of hard_sigmoid activation function. The gate predicted by STN agree well with our intuition.

Turn	Utterance, State, Gate
1	**sys**: hello , welcome to the cambridge restaurant system ? you can ask for restaurants by area , price range or food type . how may i help you ? **user**: id like a cheap restaurant in the south part of town **State**: inform(pricerange=cheap,area=south) **Gate**: food=1,pricerange=1,area=1
2	**sys**: there are 9 restaurants in the south of town . what type of food do you want ? **user**: any type of food **State**: inform(food=dontcare,pricerange=cheap,area=south) **Gate**: food=1,pricerange=0,area=0
3	**sys**: there are 9 restaurants in the south of town serving any kind of food . what price range would you like ? **user**: cheap **State**: inform(food=dontcare,pricerange=cheap,area=south) **Gate**: food=0,pricerange=1,area=0
4	**sys**: the lucky star is a nice place in the south of town and the prices are cheap **user**: can i have their phone number please **State**: inform(food=dontcare,pricerange=cheap,area=south) **Gate**: food=0,pricerange=0,area=0
5	**sys**: the phone number of the lucky star is 01223 244277 . **user**: thank you good bye **State**: inform(food=dontcare,pricerange=cheap,area=south) **Gate**: food=0,pricerange=0,area=0

tracking as a machine reading problem which read the dialog transcriptions multiple times and answer the questions about each of the slot. STN process the dialog transcriptions one time and output all of the slot values simultaneously. We perform both the soft and hard version STNs and they exceed the performance of MenN2N by a explicit margin(Table 2 half bottom). We can train the hard version STN from scratch or initialize it with the parameters of trained soft STN and train it continuously, each of which perform pretty well. In the text input condition, the hard STN slightly outperform the soft STN by two slot(Food and Price), it's because the text dialog utterance is accurate and the gate of state updater can be formulated as a deterministic hard form.

4.3 Analysis

Table 3 show a sample of DSTC2 test set whose state and gate has been predicted correctly by our STN. The soft STN and hard STN make the same predictions of state and gate. Hard gate can only choose the value of 0 or 1, soft gate lies in $[0, 1]$ and can also reach the extreme value of 0 or 1 by the use of hard_sigmoid activation function. In following analysis, we can found that the gate predicted by STN agree well with our intuition.

In the first turn, all the gates have been opened, it maybe because the initialization setup of the state updater. The value of food slot is the special 'not mentioned', so it doesn't appear in the state here. The value of pricerange slot and area slot are correctly predicted as cheap and south respectively. In the second turn, the system only ask for food type and the user answer with 'any' which means he/she doesn't care about(dontcare) food type, they don't talk about any other slot, so the only opened gate is food slot, it updates its value with 'dontcare', the gates of other slots are closed and the slots values follow the last turn. In the third turn, the user answer the system with cheap pricerange, so only the gate of pricerange slot is opened, it updates the pricerange slot with the same cheap value. In the last two turns, the user doesn't mention any slot, so all of the gates are closed, as a result, all the slot values keep unchanged.

5 Related Work

Dialog state tracking methods can roughly be divided into generative approach and discriminative approach(Williams, Raux, and Henderson 2016). Generative approach computer a distribution over possible(hidden) dialog states using a Bayesian network that relate the dialog state to the system action, the(unobserved) user action and ASR or SLU result. For computing efficiency, different factorizations of the hidden state and probability independence assumptions are always be made. However, generative models cannot easily incorporate large sets of potentially informative features from SLU, dialog history and elsewhere. To cope with such limitations, discriminative approaches to dialog state tracking compute scores for dialog states with discriminatively train conditional models which can directly incorporate a large number of features extracted from SLU, dialog history and so on, it can be optimized directly for prediction accuracy. The performance of discriminative approach is often superior than the generative one(Williams, Raux, and

Henderson 2016). Recent deep learning based models(Mrksic et al. 2017; Henderson, Thomson, and Young 2014b; 2014a) and our STNs fall into this kind of approach and have shown promising results. Our method extends the work of Mrksic et al.; Henderson, Thomson, and Young; Perez and Liu and overcome some drawbacks of theirs. Henderson, Thomson, and Young use a delexicalisation-based RNN model to track the dialog state without the module of NLU and feature engineering, but it rely on a heavy strategy known as delexicalisation whereby slots and values mentioned in the text are replaced with generic labels. Mrksic et al. propose the neural belief tracker(NBT) which doesn't rely on either the NLU module or the delexicalisation, it process each turn of the dialog utterance independently[11] and enumerate all of the candidate slot value pairs upon which the NBT model perform binary decisions(the candidate slot value is right or wrong). Perez and Liu cast dialog state tracking to machine reading problem and use the end to end muti-hop memory networks which is a powerful model for text understanding, however it should read the dialog history multiple times for predict only a single slot value. Based on these drawbacks, we propose the state tracking networks for dialog state tracking which directly model the state update mechanism until the current turn and output the probability distribution of the dialog state, STN doesn't need the NLU module, delexicalisation and feature engineering, it's input is the original user and system utterance(ASR n-best list or transcription text), output is the probability distribution of the dialog state, it is a very efficient end to end dialog state tracking model.

Our state tracking networks are inspired by Recurrent Entity Network(EntNet)(Henaff et al. 2016) which maintain and update a set of parallel memories which represent the state of the world, due to its explicit gating mechanism EntNet can perform both location and content-based read and write operations. STN extend the EntNet, but use a different gating(equation(6)) and updating(equation(8)) mechanism. STN also absorb the design ideology of GRU(Cho et al. 2014) and LSTM(Hochreiter and Schmidhuber 1997), but totally different from them with the scalar gate formulation.

6 Conclusion

In this paper we propose a novel model named state tracking networks for dialog state tracking, it accept original user/system utterance(ASR n-best list or transcriptions) as input and output the dialog state probability at each turn. One characteristic of our model is that it use explicit gate to model state update mechanism for each slot and it can be explained intuitively. Attribute to the recurrent structure of STN, when process the current turn, it can take all the dialog history information into consideration. STN track all of slots simultaneously end to end without the need of NLU module, it can be trained deterministically using standard backpropagation or stochastically using REINFORCE with the hard version gate. When evaluated on the authoritative DSTC2 dataset, STN can match the performance of state-of-the-art models, particularly, when experimented on the text dialog, STN exceed the performance of MemN2N with a sufficient margin.

[11]They use a rule based state update to track the dialog state.

References

Bordes, A., and Weston, J. 2016. Learning end-to-end goal-oriented dialog. *CoRR* abs/1605.07683.

Cho, K.; van Merrienboer, B.; Gülçehre, Ç.; Bahdanau, D.; Bougares, F.; Schwenk, H.; and Bengio, Y. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1724–1734.

Collobert, R.; Weston, J.; Bottou, L.; Karlen, M.; Kavukcuoglu, K.; and Kuksa, P. P. 2011. Natural language processing (almost) from scratch. *Journal of Machine Learning Research* 12:2493–2537.

Henaff, M.; Weston, J.; Szlam, A.; Bordes, A.; and LeCun, Y. 2016. Tracking the world state with recurrent entity networks. *CoRR* abs/1612.03969.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014. The second dialog state tracking challenge. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, 263–272.

Henderson, M.; Thomson, B.; and Young, S. J. 2014a. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *2014 IEEE Spoken Language Technology Workshop, SLT 2014, South Lake Tahoe, NV, USA, December 7-10, 2014*, 360–365.

Henderson, M.; Thomson, B.; and Young, S. J. 2014b. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA*, 292–299.

Hochreiter, S., and Schmidhuber, J. 1997. Long short-term memory. *Neural computation* 9(8):1735–1780.

Kim, Y. 2014. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*.

Kingma, D. P., and Ba, J. 2014. Adam: A method for stochastic optimization. *CoRR* abs/1412.6980.

Mrksic, N.; Séaghdha, D. Ó.; Wen, T.; Thomson, B.; and Young, S. J. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, 1777–1788.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, 1532–1543.

Perez, J., and Liu, F. 2016. Dialog state tracking, a machine reading approach using memory network. *arXiv preprint arXiv:1606.04052*.

Schulman, J.; Heess, N.; Weber, T.; and Abbeel, P. 2015. Gradient estimation using stochastic computation graphs. In *Advances in Neural Information Processing Systems 28: Annual Conference on Neural Information Processing Systems 2015, December 7-12, 2015, Montreal, Quebec, Canada*, 3528–3536.

Shang, L.; Lu, Z.; and Li, H. 2015. Neural responding machine for short-text conversation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, 1577–1586.

Vinyals, O., and Le, Q. 2015. A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Wen, T.; Gasic, M.; Mrksic, N.; Rojas-Barahona, L. M.; Su, P.; Ultes, S.; Vandyke, D.; and Young, S. J. 2016. A network-based end-to-end trainable task-oriented dialogue system. *CoRR* abs/1604.04562.

Williams, J. D., and Young, S. J. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language* 21(2):393–422.

Williams, J. D.; Raux, A.; and Henderson, M. 2016. The dialog state tracking challenge series: A review. *D&D* 7(3):4–33.

Williams, R. J. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine Learning* 8:229–256.

Zaremba, W.; Sutskever, I.; and Vinyals, O. 2014. Recurrent neural network regularization. *CoRR* abs/1409.2329.