Dialog State Tracking with Reinforced Data Augmentation

Yichun Yin,* Lifeng Shang,* Xin Jiang, Xiao Chen, Qun Liu
Huawei Noah's Ark Lab
{yinyichun, shang.lifeng, jiang.xin, chen.xiao2, qun.liu}@huawei.com

Abstract

Neural dialog state trackers are generally limited due to the lack of quantity and diversity of annotated training data. In this paper, we address this difficulty by proposing a reinforcement learning (RL) based framework for data augmentation that can generate high-quality data to improve the neural state tracker. Specifically, we introduce a novel *contextual bandit generator* to learn fine-grained augmentation policies that can generate new effective instances by choosing suitable replacements for specific context. Moreover, by alternately learning between the *generator* and the state *tracker*, we can keep refining the generative policies to generate more high-quality training data for neural state tracker. Experimental results on the WoZ and MultiWoZ (restaurant) datasets demonstrate that the proposed framework significantly improves the performance over the state-of-the-art models, especially with limited training data.

Introduction

With the increasing popularity of intelligent assistants such as Alexa, Siri and Google Duplex, the research on spoken dialog systems has gained a great deal of attention in recent years (Gao, Galley, and Li 2018). Dialog state tracking (DST) (Williams et al. 2013) is an essential component of most spoken dialog systems, aiming to track user's goal at each step in a dialog. Based on that, the dialog agent decides how to converse with the user. In a *slot-based* dialog system, the dialogue states are typically formulated as a set of slot-value pairs and one concrete example is as follows:

> **User:** Grandma wants Italian, any suggestions?
> `State: inform(food=Italian)`
> **Agent:** Would you prefer south or center?
> **User:** It doesn't matter. Whichever is less expensive.
> `State: inform(food=Italian, price=cheap, area=don't care)`

The state-of-the-art models for DST are based on neural network (Henderson, Thomson, and Young 2014; Mrkšić et al. 2017; Zhong, Xiong, and Socher 2018; Ren et al. 2018; Sharma, Choubey, and Huang 2019; Chao and Lane 2019; Wu et al. 2019). They typically predict the probabilities of the candidate slot-value pairs with the user utterance, previous system actions or other external information as inputs, and then determine the final value of each slot based on the probabilities. Although the neural network based methods are promising with advanced deep learning techniques such as gating and self-attention mechanisms (Lin et al. 2017; Vaswani et al. 2017), the data-hungry nature makes them difficult to generalize well to the scenarios with limited or sparse training data.

To alleviate the data sparsity in DST, we propose a reinforced data augmentation (RDA) framework to increase both the amount and diversity of the training data. The RDA learns to generate high-quality labeled instances, which are used to re-train the neural state trackers to achieve better performances. As shown in Figure 1, the RDA consists of two primary modules: Generator and Tracker. The two learnable modules alternately learn from each other during the training process. On one hand, the Generator module is responsible for generating new instances based on a parameterized gen-

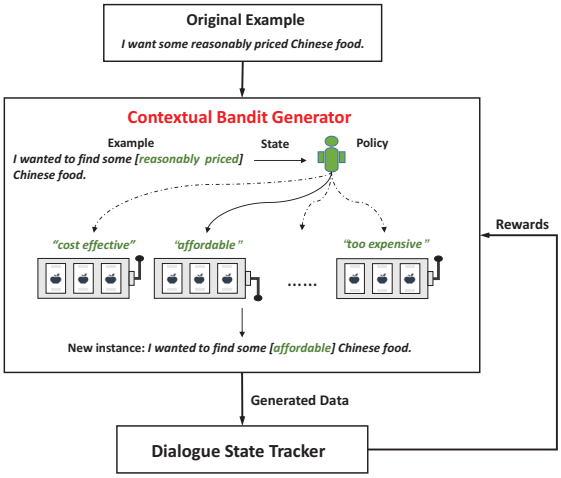

Figure 1: An overview of our framework. Given a dataset, we induce new instances using the RL-based Generator to improve the DST Tracker. The Generator is trained with the rewards from the Tracker. The learning process is performed in an alternate manner.

*Corresponding authors.

erative policy, which is trained with the rewards from the Tracker module. The Tracker, on the other hand, is refined via the newly generated instances from the Generator.

Data augmentation performs perturbation on the original dataset without actually collecting new data, which has been widely used in the field of computer vision (Krizhevsky, Sutskever, and Hinton 2012) and speech recognition (Ko et al. 2015), but relatively limited in natural language processing (Kobayashi 2018). The reason is that, in contrast to image augmentation (e.g., rotating or flipping images), it is significantly more difficult to augment text because it requires preserving the semantics and fluency of newly augmented data. In this paper, to derive a more general and effective policy for text data augmentation, we adopt a coarse-to-fine strategy to model the generation process. Specifically, we initially use some coarse-grained methods to get candidates (such as *cost effective*, *affordable* and *too expensive* in Figure 1), some of which are inevitably noisy or unreliable for the specific sentence context. We then adopt RL to learn the policies for selecting high quality candidates to generate new instances, where the total rewards are obtained from the Tracker. After learning the Generator, we use it to induce more training data to re-train the Tracker. Accordingly, the Tracker will further provide more reliable rewards to the Generator. With alternate learning, we can progressively improve the generative policies for data augmentation and at the same time learn the better Tracker with the augmented data.

To demonstrate the effectiveness of the proposed RDA framework in DST, we conduct extensive experiments with the WoZ (Wen et al. 2017) and MultiWoZ (restaurant) (Budzianowski et al. 2018) datasets. The results show that our model consistently outperforms the strong baselines and achieves new state-of-the-art results. In addition, the effects of the hyper-parameter choice on performance are analyzed and case studies on the policy network are performed.

The main contributions of this paper include:

- We propose a novel framework of data augmentation for dialog state tracking, which can generate high-quality labeled data to improve neural state trackers.
- We use RL for the Generator to produce effective text augmentation.
- We demonstrate the effectiveness of the proposed framework on two datasets, showing that the RDA can consistently boost the state-tracking performance and obtain new state-of-the-art results.

Reinforced Data Augmentation

We elaborate on our framework in three parts: the Tracker module, the Generator module, and the alternate learning algorithm.

Tracker Module

The dialog state tracker aims to track the user's goal during the dialog process. At each turn, given the user utterance and the system action/response[1], the trackers first estimate the probabilities of the candidate slot-value pairs[2], and then the pair with the maximum probability for each slot is chosen as the final prediction. To obtain the dialog state of the current turn, trackers typically use the newly predicted slot-values to update the corresponding values in the state of previous turn. One concrete example of the Tracker module is illustrated in Figure 2.

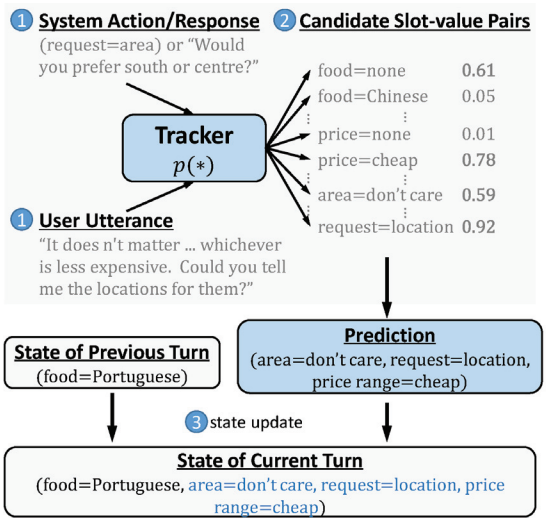

Figure 2: The Tracker module. (1) System action or response, and user utterance as input; (2) The tracker predicts the probabilities of all possible slot-value pairs; (3) The prediction and state of previous turn are used to update the state of the current turn.

Our RDA framework is generic and can be applied to different types of tracker models. To demonstrate its effectiveness, we experiment with two different trackers: the state-of-the-art GLAD (Attentive Dialogue State Tracker) model and the classical NBT-CNN (Neural Belief Tracking - Convolutional Neural Networks) model (Mrkšić et al. 2017). We use the Tracker to refer to GLAD and NBT-CNN in the following sections.

Generator Module

We formulate data augmentation as an optimal text span replacement problem in a labeled sentence. Specifically, given the tuple of a sentence $\mathbf{x}$, its label y, and the text span p of the sentence, the Generator aims to generate a new training instance $(\mathbf{x}', y')$ by substituting p in $\mathbf{x}$ with an optimal candidate p' from a set of candidates for p, which we denote as $\mathbf{C}_p$.

In the span-based data augmentation, we can replace the text span with its paraphrases derived either from existing paraphrase databases or neural paraphrase generation models e.g. (Zhao et al. 2009; Li et al. 2018). However, directly applying the coarse-grained approach can introduce ineffective or noisy instances for training, and eventually hurt the performance of trackers. Therefore, we train the Generator

[1]If the system actions do not exist in the dataset, we use the system response as the input.

[2]For each slot, *none* value is added as one candidate slot-value pair.

to learn fine-grained generation policies to further improve the quality of the augmented data.

Generation Process. The problem of high quality data generation is modeled as a *contextual bandit* (or *one-step reinforcement learning*) (Dudik et al. 2011). Formally, at each trial of a contextual bandit, the context including the sentence x and its text span p, is sampled and shown to the agent, then the agent selects a candidate p' from $\mathbf{C}_p$ to generate a new instance $\mathbf{x}'$ by replacing p with p'.

Policy Learning. The policy $\pi_\theta(\mathbf{s}, p')$ represents a probability distribution over the valid actions at the current trial, where the state vector $\mathbf{s}$ is extracted from the sentence $\mathbf{x}$, the text span p and the candidate p'. $\mathbf{C}_p$ forms the action space of the agent given the state $\mathbf{s}$, and the reward R is a scalar value function. The policy is learned to maximize the expected rewards:

$$\mathcal{J}(\theta) = E_{\pi_\theta}[R], \tag{1}$$

where the expectation is taken over state $\mathbf{s}$ and action p'.

The policy $\pi_\theta(\mathbf{s}, p')$ decides which $p' \in \mathbf{C}_p$ to take based on the state $\mathbf{s}$, which is formulated as:

$$\mathbf{s} = [\mathbf{p}, \mathbf{p}'_{emb}, \mathbf{p}'_{emb} - \mathbf{p}_{emb}, \mathbf{p}'_{emb} \circ \mathbf{p}_{emb}], \tag{2}$$

where $\mathbf{p}$ is the contextual representation of p, which is derived from the hidden states in the encoder of the Tracker, $\mathbf{p}_{emb}$ and $\mathbf{p}'_{emb}$ are the word embeddings of p and p' respectively. For multi-word phrases, we use the average representations of words as the phrase representation. We use a *two-layer fully connected network* and *sigmoid* to compute the score function $f(\mathbf{s}, p')$ of p being replaced by p'. As each p has multiple choices of replacement $\mathbf{C}_p$, we normalize the scores and obtain the final probabilities for the alternative phrases:

$$\pi_\theta(\mathbf{s}, p') = \frac{f(\mathbf{s}, p')}{\sum_{\tilde{p} \in \mathbf{C}_p} f(\mathbf{s}, \tilde{p})}. \tag{3}$$

The sampling-based *policy gradient* is used to approximate the gradient of the expected reward. To obtain more feedback and make the policy learning more stable, as illustrated in Figure 3, we propose to use a two-step sampling method: at first, sample a bag of sentences $B = \{(\mathbf{x}_i, y_i, p_i)\}_{1 \le i \le T}$, then iteratively sample a candidate ${p'}_{i,j}$ for each instance in B according to the current policy, obtaining a new bag of instances ${B'}_j = \{({\mathbf{x}'}_{i,j}, {y'}_{i,j}, {p'}_{i,j})\}_{1 \le i \le T}$. After running the bag-level sampling M times, the gradient of objective function can be estimated as:

$$\nabla \mathcal{J}(\theta) \approx \tfrac{1}{M} \sum_{j=1}^{M} \sum_{i=1}^{T} \nabla_\theta \log \pi_\theta(\mathbf{s}_{i,j}, {p'}_{i,j}) R_{i,j}, \tag{4}$$

where $\mathbf{s}_{i,j}$ and ${p'}_{i,j}$ denote the state and action of the i-th instance-level sampling from the j-th bag-level sampling, respectively. $R_{i,j}$ is the corresponding reward.

Reward Design. One key problem is assigning suitable rewards to various actions ${p'}_{i,j}$ given state $\mathbf{s}_{i,j}$. We design two kinds of rewards: bag-level reward R^{B}_j and instance-level reward $R^{\mathrm{I}}_{i,j}$ in reinforcement learning. The bag-level reward (Feng et al. 2018; Qin, Weiran, and Wang 2018) indicates whether the new sampled bag is helpful to improve the Tracker and the instances in the same bag receive the same reward value. While the instance-level reward assigns different reward values to the instances in the same sampled bags by checking whether the instance can cause the Tracker to make incorrect prediction (Kang et al. 2018; Ribeiro, Singh, and Guestrin 2018). We sum two kinds of rewards as the final reward: $R_{i,j} = R^{\mathrm{B}}_j + R^{\mathrm{I}}_{i,j}$, for more reliable policy learning.

Bag-level reward R^B_j: we re-train the Tracker with each sampled bag and use their performance (e.g., joint goal accuracy (Henderson, Thomson, and Williams 2014)) on the validation set to indicate their rewards. Suppose the performance of the j-th bag ${B'}_j$ is denoted as ${U'}_j$, the bag-level rewards are formulated as:

$$R^{\mathrm{B}}_j = \frac{2({U'}_j - \min(\{{U'}_{j^*}\}))}{\max(\{{U'}_{j^*}\}) - \min(\{{U'}_{j^*}\})} - 1, \tag{5}$$

where $\{{U'}_{j^*}\}$ refers to the set $\{{U'}_{j^*}\}_{1 \le j^* \le M}$. Here we scale the value to be bounded in the range of [-1, 1] to alleviate the instability in RL training[3].

Instance-level reward $R^{\mathrm{I}}_{i,j}$: we evaluate each generated instance $(\mathbf{x}'_{i,j}, y'_{i,j})$ in the bag and denote the instance which causes the Tracker to make wrong prediction, as *large-loss instance* (LI) (Han et al. 2018). Compared to the non-LIs, the LIs are more informative and can induce larger loss for training the Tracker. Thus, in the design of instance-level rewards, the LI is encouraged more when its corresponding bag reward is positive, and punished more when its bag reward is negative. Specifically, we define the instance-level reward as follow:

$$R^{\mathrm{I}}_{i,j} = \begin{cases} \mathrm{c}, & R^{\mathrm{B}}_j \ge 0 \wedge \mathbb{I}_{\mathrm{LI}}(\mathbf{x}'_{i,j}, y'_{i,j}) = 1 \\ \mathrm{c}/2, & R^{\mathrm{B}}_j \ge 0 \wedge \mathbb{I}_{\mathrm{LI}}(\mathbf{x}'_{i,j}, y'_{i,j}) = 0 \\ -\mathrm{c}, & R^{\mathrm{B}}_j < 0 \wedge \mathbb{I}_{\mathrm{LI}}(\mathbf{x}'_{i,j}, y'_{i,j}) = 1 \\ -\mathrm{c}/2, & R^{\mathrm{B}}_j < 0 \wedge \mathbb{I}_{\mathrm{LI}}(\mathbf{x}'_{i,j}, y'_{i,j}) = 0, \end{cases} \tag{6}$$

where $\mathbb{I}_{\mathrm{LI}}(\mathbf{x}'_{i,j}, y'_{i,j})$ is an indicator function of being a LI. We obtain the $\mathbb{I}_{\mathrm{LI}}(\mathbf{x}'_{i,j}, y'_{i,j})$ value by checking if the pre-trained Tracker can correctly predict the label on the generated example. c is a hyper-parameter, which is set to 0.5 by running a grid search over the validation set.

Alternate Learning

In the framework of RDA, the learning of Generator and Tracker is conducted in an alternate manner, which is detailed in Algorithm 1 and Figure 3.

The text span p to be replaced has different distribution in the training set. To make learning more efficient, we first sample one text span p, then sample one sentence $(\mathbf{x}, y)$ from the sentences containing p. This process is made iteratively to obtain a bag B. To learn the Generator, we generate M bags of instances by running the policy, compute their rewards and update the policy network via the policy gradient method. To learn the Tracker, we augment the training data by the updated policy. Particularly for each $(\mathbf{x}, y, p)$, we

[3] In this work, the original text span p is also used as one candidate in $\mathbf{C}_p$, which actually acts as an *implicit Baseline* (Sutton and Barto 2018) in RL training.

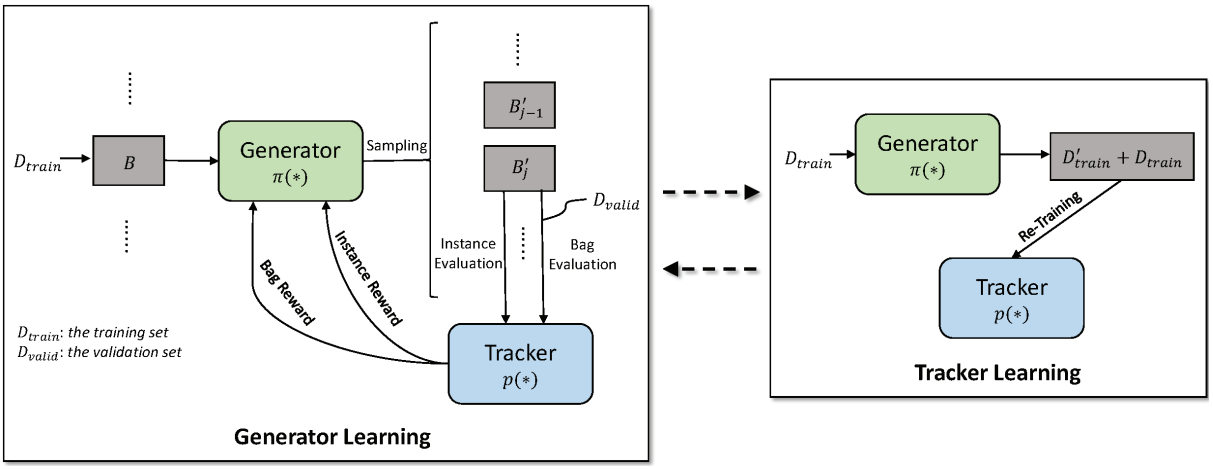

Figure 3: The algorithm flow of the reinforced data augmentation framework. The left is the Generator learning and the right is the Tracker learning. The two learning processes are performed in an alternate manner.

Algorithm 1 The Reinforced Data Augmentation

Input: Pre-trained Tracker with parameters θ_r; the randomly initialized Generator with parameters θ_π;
Output: Re-trained Tracker

Store θ_π
for $l = 1 \rightarrow L$ **do**
  Re-initialize the Generator with θ_π
  for $n = 1 \rightarrow N$ **do**
    Re-initialize the Tracker with θ_r
    Sample a bag B
    for $j = 1 \rightarrow M$ **do**
      Sample a new bag B'_j
    end for
    Compute bag reward with Eq. 5
    Compute instance reward with Eq. 6
    Update θ_π by the gradients in Eq.4
  end for
  Obtain new data D' by the Generator
  Re-train the Tracker on $D + D'$, update θ_r
end for
Save the Tracker with θ_r which performs best on the validation set among the L epochs

generate a new instance $(\mathbf{x}', y', p')$ by sampling based on the learned policies. To further reduce the effect of noisy augmented instances, we remove the new instance if its p' has minimum probability among $\mathbf{C}_p$. We randomly initialize the policy at each epoch to make the generator learn adaptively which policy is best for the current Tracker. The alternate learning is performed multiple rounds and the Tracker with the best performances on the validation set is saved.

Experiment

In this section, we show the experimental results to demonstrate the effectiveness of our framework.

Dataset and Evaluation

We use WoZ (Wen et al. 2017) and MultiWoZ (Budzianowski et al. 2018) to evaluate the proposed framework on the task of dialog state tracking[4]. Following the work (Budzianowski et al. 2018), we extract the restaurant domain of the MultiWoZ as the evaluation dataset, denoted as MultiWoZ (restaurant). Both WoZ and MultiWoZ (restaurant) are in the restaurant domain. In the experiment, we use the widely used joint goal accuracy (Henderson, Thomson, and Williams 2014) as the evaluation metric, which measures whether all slot values of the updated dialog state exactly match the ground truth values at every turn.

Implementation Details

We implement the proposed model using PyTorch[5]. All hyper-parameters of our model are tuned based on the validation set. To demonstrate the robustness of our model, we use the similar hyper-parameter settings for both datasets. Following the previous work (Ren et al. 2018; Zhong, Xiong, and Socher 2018), we concatenate the pre-trained GloVe embeddings (Pennington, Socher, and Manning 2014) and the character embeddings (Hashimoto et al. 2017) as the final word embeddings and keep them fixed when training. The epoch number of the alternate learning L, the epoch number of the generator learning N and the sampling times M for each bag are set to 5, 200 and 2 respectively. We set the dimensions of all hidden states to 200 in both the Tracker and the Generator, and set the head number of multi-head Self-Attention to 4 in the Tracker. All learnable parameters are optimized by the ADAM optimizer with a learning rate of 1e-3. The batch size is set to 16 in the Tracker learning, and the bag size in the Generator learning is set to 25.

[4]DSTC2 (Mrkšić et al. 2017) dataset is not used because its clean version (http://mi.eng.cam.ac.uk/~nm480/dstc2-clean.zip) is no longer available.

[5]https://pytorch.org/

Model	WoZ	Multi
Delexicalised Model	70.8	71.2
NBT-DNN	84.4	80.3
NBTKS	85.5	80.9
StateNet	88.9	82.4
GCE	88.5	83.5
NBT-CNN	84.0 ±0.6	79.8 ±1.0
+ BT	82.7 ±0.6	75.6 ±0.9
+ DA	84.2 ±0.5	79.7 ±0.7
+ **RDA**	87.9‡±0.3	83.4†±0.6
GLAD	88.3 ±0.3	83.6 ±0.9
+ BT	86.6 ±0.3	79.0 ±1.0
+ DA	88.0 ±0.5	82.7 ±0.7
+ **RDA**	**90.7**‡**±0.2**	**86.7**†**±0.5**

Table 1: Comparison of our model and other baselines. BT is the back-translation based the data augmentation, DA refers the coarse-grained data augmentation without the reinforced framework, and Multi refers the dataset MultiWoZ (restaurant). t-test is conducted in our proposed models and original trackers (NBT-CNN and GLAD) are used as the comparison baselines. † and ‡: significant over the baseline trackers at 0.05/0.01. The mean and the standard deviation are also reported.

To avoid over-fitting, we apply dropout to the layer of word embeddings with a rate of 0.2. We also assign rewards based on subsampled validation set with a ratio of 0.3 to avoid over-fitting the policy network on the validation set.

In our experiments, the newly augmented dataset is n times the size of the original training data ($n = 5$ for the Woz and $n = 3$ for MultiWoz). At each iteration, we randomly sample a subset of the augmented data to train the Tracker. The sampling ratios are 0.4 for Woz and 0.3 for MutiWoz.

For the coarse-grained data augmentation method, we have tried the current neural paraphrase generation model. The preliminary experiment indicates that almost all generated sentences are not helpful for the task of DST. The reason is that most of the neural paraphrase generation models require additional labeled paraphrase corpus which may not be always available (Ray, Shen, and Jin 2018). In this work, we extract unigrams, bigrams and trigrams in the training data as the text spans in the generation process. After that, we retrieve the paraphrases for each text span from the PPDB[6] database as the candidates. We also use the golden slot value in the sentence as the text spans, the other values of the same slot as the candidates and the label will be changed accordingly.

Baseline Methods

We compare our model with some baselines. **Delexicalised Model** uses generic tags to replace the slot values and employs a CNN for turn-level feature extraction and a Jordan RNN for state updates (Henderson, Thomson, and Young 2014; Wen et al. 2017). **NBT-DNN** and **NBT-CNN** respectively use the summation and convolution filters to learn the representations for the user utterance, candidate slot-value pair and the system actions (Mrkšić et al. 2017). Then, they fuse these representations by a gating mechanism for the final prediction. **NBTKS** has a similar structure to NBT-DNN and NBT-CNN, but with a more complicated gating mechanism (Ramadan, Budzianowski, and Gasic 2018). **StateNet** learns a representation from the dialog history, and then compares the distances between the learned representation and the vectors of the candidate slot-value pairs for the final prediction (Ren et al. 2018). **GLAD** is a global-locally self-attentive state tracker, which learns representations of the user utterance and previous system actions with global-local modules (Zhong, Xiong, and Socher 2018). **GCE** is developed based on GLAD by using global recurrent networks rather than the global-local modules (Nouri and Hosseini-Asl 2018).

We use the coarse-grained data augmentation (**DA**) without the reinforced framework, which is described at the implementation section, as the baseline. The paraphrasing method is widely used for the data augmentation in the previous work (Ray, Shen, and Jin 2018; Hou et al. 2018), thus we also use back-translation (**BT**) based paraphrasing as the baseline. Both the **DA** and **BT** use the same amount of augmented instances with the proposed reinforced data augmentation (**RDA**) to ensure a fair comparison.

[6]http://paraphrase.org/

Dataset	Model	10%	20%	50%
WoZ	GLAD	50.1	72.5	81.7
	+ RDA	**66.8**	**81.5**	**86.9**
Multi	GLAD	60.0	72.6	77.6
	+ RDA	**71.5**	**81.2**	**85.2**

Table 2: The results with different sub-sampling ratios on WoZ and MultiWoZ (restaurant).

Setting	WoZ	Multi
RDA	**90.7**	**86.7**
- Bag Reward	89.1	84.3
- Instance Reward	89.8	85.4
DA	88.0	82.7

Table 3: Ablation study of performances on the test set of WoZ and MultiWoZ.

Results and Analyses

We compare our model with baselines and the joint goal accuracy is used as the evaluation metric. The results are shown in Table 1.

From the table, we observe that the proposed GLAD achieves comparable performances (88.3% and 83.6%) with other state-of-the-art models on both datasets. The framework RDA can further boost the performances of the competitive GLAD by the margin of 2.4% and 3.1% on two datasets respectively, achieving new state-of-the-art results (90.7% and 86.7%). Compared with the GLAD, the classical

NBT-CNN with the RDA framework obtains more improvements: 3.9% and 3.6%. We also conduct significance test (t-test), and the results show that the proposed RDA achieves significant improvements over baseline models ($p < 0.01$ and $p < 0.05$ respectively for WoZ and MultiWoZ (restaurant)).

For the back-translation based augmentation (BT), it consistently decreases the performances of NBT-CNN and GLAD in both datasets. The reason is that the some augmented instances generated by the end-to-end method are noisy and the semantic of original text is shifted. The noisy instances are fed into the neural networks and hurt their performances.

The table also shows that directly using coarse-grained data augmentation methods without the RDA is less effective, and can even degrade the performances, as it may generate noisy instances. The results show that: using the RDA, the GLAD achieves improvements of (88.0%→90.7%) and (82.7%→86.7%) respectively on the WoZ and MultiWoZ. The NBT-CNN obtains improvements of (84.2%→87.9%) and (79.7%→83.4%) respectively. Overall, the results indicate that the RDA framework offers an effective mechanism to improve the quality of augmented data.

To further verify the effectiveness of the RDA when the training data is scarce, we conduct sub-sampling experiments with the GLAD tracker trained on different ratios [10%, 20%, 50%] of the training set. The results on both datasets are shown in Table 2. We find that our proposed RDA methods consistently improve the original tracker performance. Notably, we obtain ∼10% improvements with [10%, 20%] ratios of training set on both WoZ and MultiWoZ (restaurant), which indicates that the RDA framework is particularly useful when the training data is limited.

To evaluate the performance of different level rewards, we perform ablation study with GLAD on both the WoZ and MultiWoz datasets. The results are shown in Table 3. From the table we can see that both rewards can provide the improvements of 1% to 2% in the datasets and the bag-level reward achieves larger gains than the instance-level reward. Compared with DA setting, RDA obtains the improvements of 3% to 4% on the datasets by combining the both rewards, which indicates that the summation reward is more reliable for policy learning than individual ones.

Effects of Hyper-parameters

In this subsection, we investigate the effects of the number of newly augmented data in the Tracker learning, the epoch number of the alternate learning L and the epoch number of the Generator learning N on performance. We conduct experiments with the GLAD tracker which is evaluated on the validation set of WoZ and the joint goal accuracy is used as the evaluation metric.

Number of newly augmented data: we use 0 to 5 times the size of original data in the Tracker learning. The performance is shown in Figure 4 (top). The model continues to improve when the number of newly added examples is less than 2 times the original data. When we add more than twice the amount of original data, the improvement is not significant.

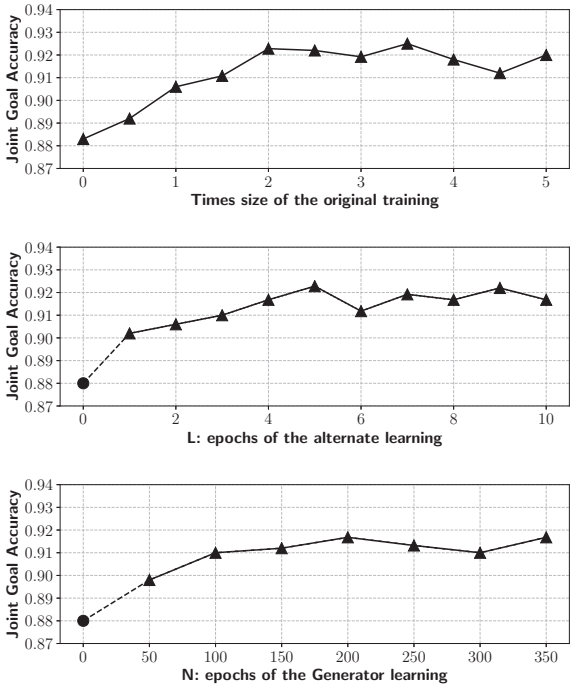

Figure 4: Results of different hyper-parameters. Top: different times the size of original data; Middle: different epochs of alternate learning; Bottom: different epochs of the Generator learning. The solid circles of $L = 0$ and $N = 0$ in the figure refer to the model of coarse-grained data augmentation (DA).

Epoch number of the alternate learning: we vary L from 0 to 10 and the performance is shown in Figure 4 (middle). We can see that, with alternate learning, the model continues to improve when $L \leq 5$, and becomes stable with no improvement after $L > 5$.

Epoch number of the Generator learning: we vary N from 0 to 350, and the performance is shown in Figure 4 (bottom). We find that the performance increases dramatically when $N \leq 200$, and shows no improvement after $N > 200$. It shows that the Generator needs a large N to ensure a good policy.

Case Study for Policy Network

We sample four sentences from WoZ to demonstrate the effectiveness of the Generator policy in the case study. Due to limited space, we present the candidate phrases with maximum and minimum probabilities derived from the policy network and the details are shown in Table 4.

We observe that both high-quality and low-quality replacements exist in the candidate set. The high-quality replacements will generate reliable instances, which can potentially improve the generalization ability of the Tracker. The low-quality ones will induce noisy instances and can reduce the performance of the Tracker. From the results of the policy network, we find that our Generator can automatically infer the quality of candidate replacements, assigning higher probabilities to the high-quality candidates and lower

probabilities to the low-quality candidates.

Sentence $\mathbf{x}$ and text span p	Candidates $\mathbf{C}_p$
Thanks , [**could you give**] me the phone number for the restaurant?	i was wonder if you could provide are you able to
What restaurants are on the east side that are not [**overpriced**] ?	too expensive cheap enough
What is a affordable restaurant in the [**south side part**] of town?	south end southern countries
I want Cuban food and i [**do n't care**] about the price range.	do n't worry do n't give a danm

Table 4: Case study for the Generator policy. The phrases with maximum policy values are listed at the first line in each cell of Candidates $\mathbf{C}_p$ and the ones with minimum values are listed at the second line.

Related Work

Dialog State Tracking. DST is studied extensively in the literature (Williams, Raux, and Henderson 2016). The methods can be classified into three categories: rue-based, generative (DeVault and Stone 2007; Williams 2008), and discriminative (Metallinou, Bohus, and Williams 2013) methods. The discriminative methods (Metallinou, Bohus, and Williams 2013) study dialog state tracking as a classification problem, designing a large number of features and optimizing the model parameters by the annotated data. Recently, neural networks based models with different architectures have been applied in DST (Henderson, Thomson, and Young 2014; Zhong, Xiong, and Socher 2018). These models initially employ CNN (Wen et al. 2017), RNN (Ramadan, Budzianowski, and Gasic 2018), self-attention (Nouri and Hosseini-Asl 2018) to learn the representations for the user utterance and the system actions/response, then various gating mechanisms (Ramadan, Budzianowski, and Gasic 2018) are used to fuse the learned representations for prediction. Another difference among these neural models is the way of parameter sharing, most of which use one shared global encoder for representation learning, while the work (Zhong, Xiong, and Socher 2018) pairs each slot with a local encoder in addition to one shared global encoder. Although these neural network based trackers obtain state-of-the-art results, they are still limited by insufficient amount and diversity of annotated data. To address this difficulty, we propose a method of data augmentation to improve neural state trackers by adding high-quality generated instances as new training data.

Data Augmentation. Data augmentation aims to generate new training data by conducting transformations (e.g. rotating or flipping images, audio perturbation, etc.) on existing data. It has been widely used in computer vision (Krizhevsky, Sutskever, and Hinton 2012) and speech recognition (Ko et al. 2015). In contrast to image or speech transformations, it is difficult to obtain effective transformation rules for text which can preserve the fluency and coherence of newly generated text and be useful for specific tasks. There is prior work on data augmentation in NLP (Hou et al. 2018; Ray, Shen, and Jin 2018). These approaches do not specially design some mechanisms to filter out low-quality generated instances. In contrast, we propose a coarse-to-fine strategy for data augmentation, where the fine-grained generative polices learned by RL are used to automatically reduce the noisy instances and retain the effective ones.

Reinforcement Learning in NLP. RL is a general purpose framework for decision making and has been applied in many NLP tasks such as relational reasoning (Xiong, Hoang, and Wang 2017), sequence learning (Ranzato et al. 2015; Li et al. 2018; Celikyilmaz et al. 2018), summarization (Paulus, Xiong, and Socher 2017; Dong et al. 2018), text classification (Wu, Li, and Wang 2018; Feng et al. 2018) and dialog (Singh et al. 2000; Li et al. 2016). Previous works by (Feng et al. 2018) designs RL algorithm to learn how to filter out noisy ones. Our work is significantly different from these works, especially in the problem settings and model frameworks. The previous work assume there are many distant sentences. However, in our work we only know possible replacements, and our RL algorithm should learn how to choose optimal replacements to "generate" new high-quality sentences. Moreover, the action space and reward design are different.

Conclusion and Future Work

We have proposed a reinforced data augmentation (RDA) method for dialogue state tracking in order to improve its performance by generating high-quality training data. The Generator and the Tracker are learned in an alternate manner, i.e. the Generator is learned based on rewards from the Tracker while the Tracker is re-trained and boosted with the new high-quality data augmented by the Generator. We conducted extensive experiments on the datasets of WoZ and MultiWoZ (restaurant); the results demonstrate the effectiveness of our framework. In future work, we would conduct experiments on more NLP tasks and introduce neural network based paraphrasing method in the RDA framework.

Acknowledgments

We thank Milan Gritta for his fruitful discussions and all anonymous reviewer for their insightful comments.

References

Budzianowski, P.; Wen, T.-H.; Tseng, B.-H.; Casanueva, I.; Ultes, S.; Ramadan, O.; and Gasic, M. 2018. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*.

Celikyilmaz, A.; Bosselut, A.; He, X.; and Choi, Y. 2018. Deep communicating agents for abstractive summarization. In *NAACL*, volume 1.

Chao, G.-L., and Lane, I. 2019. Bert-dst: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. *arXiv preprint arXiv:1907.03040*.

DeVault, D., and Stone, M. 2007. Managing ambiguities across utterances in dialogue. In *Decalog*.

Dong, Y.; Shen, Y.; Crawford, E.; van Hoof, H.; and Cheung, J. C. K. 2018. Banditsum: Extractive summarization as a contextual bandit. In *EMNLP*.

Dudik, M.; Hsu, D.; Kale, S.; Karampatziakis, N.; Langford, J.; Reyzin, L.; and Zhang, T. 2011. Efficient optimal learning for contextual bandits. *arXiv preprint arXiv:1106.2369*.

Feng, J.; Huang, M.; Zhao, L.; Yang, Y.; and Zhu, X. 2018. Reinforcement learning for relation classification from noisy data.

Gao, J.; Galley, M.; and Li, L. 2018. Neural approaches to conversational ai. *arXiv preprint arXiv:1809.08267*.

Han, B.; Yao, Q.; Yu, X.; Niu, G.; Xu, M.; Hu, W.; Tsang, I.; and Sugiyama, M. 2018. Co-teaching: Robust training of deep neural networks with extremely noisy labels. In *NeurIPS*.

Hashimoto, K.; xiong, c.; Tsuruoka, Y.; and Socher, R. 2017. A joint many-task model: Growing a neural network for multiple nlp tasks. In *EMNLP*.

Henderson, M.; Thomson, B.; and Williams, J. D. 2014. The second dialog state tracking challenge. In *SIGDIAL*.

Henderson, M.; Thomson, B.; and Young, S. 2014. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*.

Hou, Y.; Liu, Y.; Che, W.; and Liu, T. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *COLING*.

Kang, D.; Khot, T.; Sabharwal, A.; and Hovy, E. 2018. Adversarial training for textual entailment with knowledge-guided examples. In *ACL*.

Ko, T.; Peddinti, V.; Povey, D.; and Khudanpur, S. 2015. Audio augmentation for speech recognition. In *Interspeech*.

Kobayashi, S. 2018. Contextual augmentation: Data augmentation by words with paradigmatic relations. In *NAACL*.

Krizhevsky, A.; Sutskever, I.; and Hinton, G. E. 2012. Imagenet classification with deep convolutional neural networks. In *NeurIPS*.

Li, J.; Monroe, W.; Ritter, A.; Jurafsky, D.; Galley, M.; and Gao, J. 2016. Deep reinforcement learning for dialogue generation. In *EMNLP*.

Li, Z.; Jiang, X.; Shang, L.; and Li, H. 2018. Paraphrase generation with deep reinforcement learning. In *EMNLP*.

Lin, Z.; Feng, M.; Santos, C. N. d.; Yu, M.; Xiang, B.; Zhou, B.; and Bengio, Y. 2017. A structured self-attentive sentence embedding. *ICLR*.

Metallinou, A.; Bohus, D.; and Williams, J. 2013. Discriminative state tracking for spoken dialog systems. In *ACL*.

Mrkšić, N.; Séaghdha, D. Ó.; Wen, T.-H.; Thomson, B.; and Young, S. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*.

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

Paulus, R.; Xiong, C.; and Socher, R. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Pennington, J.; Socher, R.; and Manning, C. D. 2014. Glove: Global vectors for word representation. In *EMNLP*.

Qin, P.; Weiran, X.; and Wang, W. Y. 2018. Robust distant supervision relation extraction via deep reinforcement learning. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Ramadan, O.; Budzianowski, P.; and Gasic, M. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *ACL*.

Ranzato, M.; Chopra, S.; Auli, M.; and Zaremba, W. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Ray, A.; Shen, Y.; and Jin, H. 2018. Robust spoken language understanding via paraphrasing. *arXiv preprint arXiv:1809.06444*.

Ren, L.; Xie, K.; Chen, L.; and Yu, K. 2018. Towards universal dialogue state tracking. In *EMNLP*.

Ribeiro, M. T.; Singh, S.; and Guestrin, C. 2018. Semantically equivalent adversarial rules for debugging NLP models. In *ACL*.

Sharma, S.; Choubey, P. K.; and Huang, R. 2019. Improving dialogue state tracking by discerning the relevant context.

Singh, S. P.; Kearns, M. J.; Litman, D. J.; and Walker, M. A. 2000. Reinforcement learning for spoken dialogue systems. In *NeurIPS*.

Sutton, R. S., and Barto, A. G. 2018. *Reinforcement learning: An introduction*. MIT press.

Vaswani, A.; Shazeer, N.; Parmar, N.; Uszkoreit, J.; Jones, L.; Gomez, A. N.; Kaiser, Ł.; and Polosukhin, I. 2017. Attention is all you need. In *NeurIPS*.

Wen, T.-H.; Vandyke, D.; Mrkšić, N.; Gasic, M.; Barahona, L. M. R.; Su, P.-H.; Ultes, S.; and Young, S. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*.

Williams, J.; Raux, A.; Ramachandran, D.; and Black, A. 2013. The dialog state tracking challenge. In *SIGDIAL*.

Williams, J.; Raux, A.; and Henderson, M. 2016. The dialog state tracking challenge series: A review. *Dialogue & Discourse*.

Williams, J. D. 2008. Exploiting the asr n-best by tracking multiple dialog state hypotheses. In *Interspeech*.

Wu, C.-S.; Madotto, A.; Hosseini-Asl, E.; Xiong, C.; Socher, R.; and Fung, P. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. *arXiv preprint arXiv:1905.08743*.

Wu, J.; Li, L.; and Wang, W. Y. 2018. Reinforced co-training. In *NAACL*.

Xiong, W.; Hoang, T.; and Wang, W. Y. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*.

Zhao, S.; Lan, X.; Liu, T.; and Li, S. 2009. Application-driven statistical paraphrase generation. In *ACL*.

Zhong, V.; Xiong, C.; and Socher, R. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *ACL*.