Data Augmentation for Copy-Mechanism in Dialogue State Tracking

Xiaohui Song[1,2], Liangjun Zang[1,2(✉)], and Songlin Hu[1,2]

[1] Institute of Information Engineering, Chinese Academy of Sciences, Beijing, China
[2] School of Cyber Security, University of Chinese Academy of Sciences, Beijing, China
[3] {songxiaohui,zangliangjun,husonglin}@iie.ac.cn

Abstract. Traditional dialogue state tracking (DST) approaches need a predefined ontology to provide candidate values for each slot. To handle unseen slot values, the copy-mechanism has been widely used in DST models recently, which copies slot values from user utterance directly. Even though the state-of-the-art approaches have shown a promising performance on several benchmarks, there is still a significant gap between seen slot values (values that occur in both training set and test set) and unseen ones (values that only occur in the test set). In this paper, we aim to find out the factors that influence the generalization capability of the copy-mechanism based DST model. Our key observations include two points: 1) performance on unseen values is positively related to the diversity of slot values in the training set; 2) randomly generated strings can enhance the diversity of slot values as well as real values. Based on these observations, an interactive data augmentation algorithm is proposed to train copy-mechanism models, which augments the input dataset by duplicating user utterances and replacing the real slot values with randomly generated strings. Experimental results on three widely used datasets: WoZ 2.0, DSTC2 and Multi-WoZ demonstrate the effectiveness of our approach.

Keywords: Data Augmentation · Dialogue State Tracking · Copy-Mechanism.

1 Introduction

A task-oriented dialogue system interacts with users in natural language to accomplish tasks such as restaurant reservation or flight booking. The goal of dialogue state tracking is to provide a compact representation of the conversation at each dialogue turn, called *dialogue state*, for the system to decide the next action to take. A typical dialogue state consists of goal of user, action of the current user utterance (`inform`, `request` etc.) and dialogue history [19]. All of them are defined in a particularly designed *ontology* that restricts which slots the system can handle, and the range of values each slot can take. Tracking the goal of user is the focus of this task. To accomplish the tracking task, most DST models take the user's utterance at the current turn, a slot to track and dialogue

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

history as input, and then output the corresponding value if the user triggers the input slot. Considering an example of restaurant reservation, users can *inform* the system some restrictions of their goals (*e.g.*, `inform(food = thai)`) or *request* further information (*e.g.*, `request(phone number)`) at each turn.

USER:I'm looking for a restaurant that serves thai food.
`state:{inform(food=thai)}`
SYSTEM:There are two, one in the west end and one in the centre of town.
USER:The one on the west end, please. Can I have the phone number?
`state:{inform(food=thai, area=west),request(phone number)}`

Having access to an ontology that contains all possible values simplifies the tracking problem in many ways. However, in a real-world dialogue system, it is often impossible to enumerate all possible values for each slot. To reduce the dependence on the ontology, PtrNet[18] uses the Pointer Network[14] to handle the unknown values that are not defined in the ontology. Since then, the attention-based ***copy-mechanism*** inspired by Pointer Network have been widely used in state-of-the-art DST approaches[12, 17, 4]. The copy-mechanism based DST models directly copy slot values from the dialogue history, thus reducing the need to pre-define all slot values in the ontology.

Nonetheless, there is still a significant performance gap between seen slot values and unseen ones(*i.e.*, the values that occur in test set but not in train set), we assume the insufficient diversity of values (*i.e.*, the number of unique slot values) is the primary reason for poor generalization capability. We conduct two experiments to prove our assumption and illustrate our observations. In the first experiment, we construct synthetic test sets that only contain unseen values. For WoZ and DSTC2, we replace all values of slot `food` with food names collected from Wikipedia[4]. For Multi-WoZ dataset, we use 13 slots that contain non-enumerable values. Since it is costly to collect enough values for each slot, we use random strings to construct synthetic test set. After training the baseline model on the original training sets, we observed a huge performance gap between the original test set and the synthetic test set. In the second experiment, we augment the WoZ training sets by duplicating user utterances 10 times and replacing all slot values with collected food names. Results on the synthetic test sets show that the generalization performance is positively related to the diversity of slot values, which further confirms our conjecture.

In a real-world dialogue system, it is impractical to obtain lots of real values for each slot when attempting to improve the diversity. In order to find a more convenient way, we design the third experiment. We first use randomly generated strings as values to enhance the diversity of slot values in the WoZ training set, then train the copy-mechanism based model and test on the test set that contains real slot values. Experimental results illustrate that random strings work as

[4] `https://en.m.wikipedia.org/wiki/Lists_of_prepared_foods`

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

well as collected real values and they help a lot to get a better generalization performance.

To design a data augmentation algorithm based on the above observations, a question remains to answer: "How many values we need to generate to obtain a cost efficient result?". It is hard to answer this question in the data pre-processing phase before training (as what traditional data augmentation algorithms do), so we design our algorithm to work in the training phase. The algorithm samples user utterances iteratively from the input dataset, then replaces the real slot values with random strings, and stop training with early-stop mechanism. Experimental results show that our algorithm produces a satisfactory result at a low cost.

The rest of this paper is organized as follows: we first review the recent advances in both DST and the copy mechanism in Section 2. Then, we describe the datasets we used and the baseline model in Section 3. In Section 4, our data analysis process and key observations are described. We propose our data augmentation approach to DST task and evaluate its performance in Section 5. Finally, we conclude our work and discuss the future directions.

2 Related Works

2.1 Dialogue State Tracking

Recently, deep-learning has shown its power to the dialogue state tracking challenges [16, 5, 6]. Some approaches rely on the value set provided by ontology: Neural Belief Tracker (NBT) [9] applied representation learning to learn features as opposed to hand-crafting features; GLAD [20] addressed the problem of extraction of rare slot-value pairs; [10] tried to share parameters across slots, but the model had to iterate all slots and values defined in the ontology at each dialogue turn; [11] generated a fixed set of candidate values using a separate SLU module but suffered from error accumulation. As for those models with generalization capabilities, PtrNet [18] aimed to handle unknown values, which is the first attempt to introduce the copy mechanism into DST; TRADE[17] was a simple copy-augmented generative model that tracked dialogue states without ontology and enabled zero-shot and few-shot DST in a new domain; [4] used the pretrained language model *BERT*[3] and copy-mechanism to predict explicitly expressed values. But the generalization performance of these models still has much room for improvement.

2.2 Copy-Mechanism

Copy-mechanism in deep learning is a generic concept, which means an output is copied from an input sequence. This idea was first proposed in Pointer Network[14]. The Pointer Network is designed to learn the conditional probability of an output sequence with elements that are discrete tokens corresponding to positions in an input sequence. It can address the problems such as sorting

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

variable sized sequences and various combinatorial optimization. As mentioned in the introduction, [18] first reformulated DST problem to take advantage of the flexibility enabled by Pointer Network. Then, the copy-mechanism become a common sub-structure of the DST models, which is used in several state-of-the-art DST models such as TRADE[17], COMER[12], etc. DSTRead[4] used a span prediction method proposed in DrQA[2] for machine reading task, which is also an application of the copy mechanism.

3 Datasets, Baseline Model and Evaluation

In this section, we first describe the datasets we used, then present a baseline model that implements the copy mechanism widely used in DST models, and finally present the evaluation metrics.

3.1 Datasets

Our datasets are extracted from three datasets widely used in DST tasks, *i.e.*, WoZ 2.0[15], DSTC2[5] and Multi-WoZ 2.0[1]. To evaluate the performance of copy mechanism on unseen slot values, we focus on the slots of which the values are non-enumerable. Table 1 lists the slots, the number of slot values, and the number of samples in the following experiments.

datasets	slots	values in train(total)	data samples in train(test)
WoZ-sub	food	73(75)	2536(1646)
DSTC2-sub	food	72(73)	11677(9890)
Multi-WoZ-sub	hotel-name	35(37)	60027(73720)
	train-destination	19(20)	
	train-departure	23(25)	
	attraction-name	88(95)	
	taxi-destination	198(214)	
	taxi-departure	188(210)	
	restaurant-name	131(139)	
	restaurant-food	94(95)	
	bus-departure	1(1)	
	bus-destination	1(1)	

Table 1. Slots we use in experiments in WoZ 2.0, DSTC2 and Multi-WoZ datasets. Samples include negative samples for the slot gate (a binary classifier).

Each sample corresponds to one dialogue turn, along with a **slot**, its **active** state, and the corresponding **value**. An example is as follows:

> **utterance**:I'm looking for a restaurant that serves thai food.
> **slot**: `food`, **active**:`True`, **value**: `thai`

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

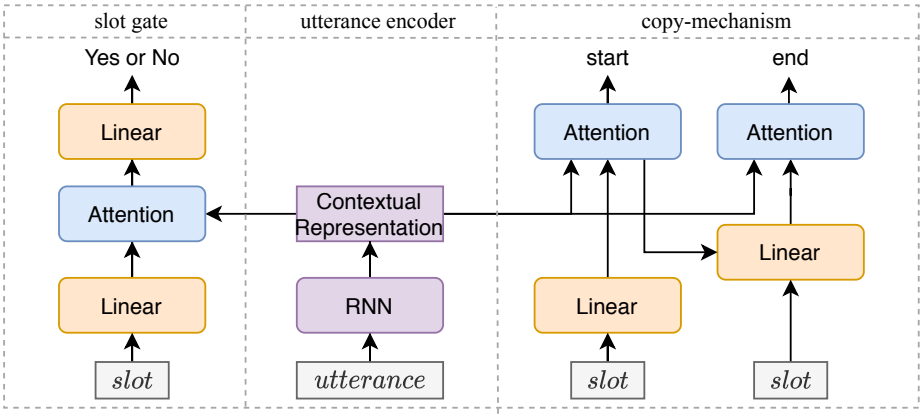

Fig. 1. The architecture of the **baseline** model. The inputs are embedding of slot, word embeddings of user and system utterance($W_1, W_2, \cdots, W_n$). It is a common sub-structure in several SOTA DST models.

The active attribute of a slot will be set to `True` (corresponding to a positive sample) if the user triggers the slot, otherwise it will be set to `False` (corresponding to a negative sample). In the test set, each turn of dialogue will be paired with each slot to reach the convincing results.

As shown in Table 1, there are a lot of overlapping values over slots between the training set and test set. To observe the performance on unseen slot values, we construct new develop datasets D'_{woz} and D'_{dstc2}, and new tests set T'_{woz} and T'_{dstc2} by replacing all values with collected real values, and guarantee that these values will not appear in any training set in this paper. As for the Multi-WoZ dataset, it is hard to collect so many values for each slots, so the synthetic develop set D'_{multi} and test set T'_{multi} for Multi-WoZ are constructed using random strings.

3.2 Baseline Model

Without loss of generality, we implement a basic copy-mechanism based model. The model takes an utterance U and a slot s as input, and output whether s is active and the positions of its corresponding value if s is active. The model consists of three important components: `utterance encoder`, `attention-based copy-mechanism`, and `slot gate`. The architecture of the baseline model is shown in Figure 1.

Utterance Encoder For utterance U, we simply concatenate user and system utterance (U^{usr} and U^{sys}) by a particular symbol <usr>.

$$U = U^{sys} \oplus \texttt{<usr>} \oplus U^{usr}, \tag{1}$$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

We use a bidirectional LSTM [7] to get the contextual representation H^P of U.

$$H^P = \text{BiLSIM}^P(U_{emb}) \in \mathbb{R}^{n \times d_h}. \tag{2}$$

where n denotes the total number of words in U, $U_{emb} \in \mathbb{R}^{n \times d_{emb}}$ is the word embeddings of U, d_{emb} is the dimension of embeddings, and d_h the dimension of LSTM hidden states.

Attention-based Copy-Mechanism We define an attention function $Attn(Q, V)$ to calculate attention scores between query feature $Q \in \mathbb{R}^{d_{emb}}$ and context feature $V \in \mathbb{R}^{n \times d_h}$. The computing process is as follows: 1) do a linear transform for both Q and V, 2) use the dot product result as attention scores, 3) normalize through `softmax` function.

$$Q' = QW_q, V' = VW_v, \alpha_i = Q'V_i', \tag{3}$$

$$scores_i = \exp \alpha_i / \sum_j^n \exp \alpha_j, \tag{4}$$

$$contexts = \sum_i^n scores_i V_i'. \tag{5}$$

We use a single linear layer ($\text{Linear}(X) = WX+b$) to encode slots embeddings s_{emb} into s_{enc}, and then use the attention function defined above to calculate both start and end positions of its value.

$$s_{enc} = \text{Linear}_{slot}(s_{emb}), \tag{6}$$

$$p_{enc} = \text{Linear}_p(s_{enc}), \tag{7}$$

$$contexts^p, scores^p = Attn_{span}(p_{enc}, H^P), \tag{8}$$

$$q_{enc} = \text{Linear}_q(s_{enc} \oplus contexts^p), \tag{9}$$

$$contexts^q, scores^q = Attn_{span}(q_{enc}, H^P), \tag{10}$$

$$start = \arg\max_j scores_j^p, \tag{11}$$

$$end = \arg\max_j scores_j^q. \tag{12}$$

Slot Gate A binary classifier is used to determine whether a slot is triggered by a user or not, where the single linear layer Linear_{cls2} produces a probability over $[0, 1]$ based on attention contexts. A slot is triggered if $cls_{result} > 0.5$.

$$s_{cls} = \text{Linear}_{cls1}(s_{emb}), \tag{13}$$

$$contexts^{cls}, scores^{cls} = Attn_{cls}(s_{cls}, H^P), \tag{14}$$

$$\alpha_{cls} = \text{Linear}_{cls2}(contexts^{cls}), \tag{15}$$

$$cls_{result} = \text{sigmoid}(\alpha_{cls}). \tag{16}$$

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

Implementation Details To enhance the effectiveness of the experiments, all experiments in this paper share the same settings. We use randomly initialized word embeddings of dimension 300 with dropout[13] rate 0.5. The model is trained with Adam[8] optimizer and the learning rate is 1e-4. We evaluate the model on the dev set and save the checkpoint and select the best to get the final result on the test set when the training process completes. All experiments are conducted on a single NVIDIA RTX 2080Ti GPU.

3.3 Evaluation Metric

We use F1 scores as the primary metric in all experiments in this paper. Specifically, if a slot is determined to be active by the slot gate and the model predicts the correct value (both the start and end positions) for the slot, then it is a true positive sample. If the slot gate outputs `False`, then the extracted value will be ignored.

4 Data Analysis and Observations

In this Section, we will analyze the factors that influence the generalization performance of the baseline model.

4.1 Original datasets may mislead the model

As shown in TRADE [17], the slots that share similar values or have correlated values have similar embeddings. We suppose that the original datasets incline the attention based copy-mechanism model to memorize values that appear in the train set rather than infer them from contexts, which may lead to a significant performance gap between seen and unseen values. To verify our argument, we train the baseline model and test on original test set and synthetic test set mentioned in Section 3.1, which contains all unseen values. The results are shown in Table 2.

datasets	original test set	synthetic test set
WoZ-sub	0.9153	0.0820
DSTC2-sub	0.9850	0.0328
Multi-WoZ-sub	0.9297	0.3607

Table 2. F1 scores on two datasets. The synthetic test sets of WoZ and DSTC2 contains all unseen values collected from Wikipedia, and that of Multi-WoZ is constructed using random strings.

As shown in Table 2, the model performs well on original test sets but poorly on synthetic test sets. The only difference between two kinds of test sets is the values set: slot values in the original test sets and the training sets have a great

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

overlap but the synthetic test sets do not. The low diversity of slot values and strong correlation with the outputs might mislead the model. The model might mistakenly think that the values are the crucial feature and overfits on them, so the model fails when it comes to an unobserved value.

4.2 Greater diversity of values brings better generalization

We have argued that the model overfits on the values when there are few values in the training set. In other words, the model pays little attention to context information of slot values, which is a very useful feature for the model to recognize the correct value for a slot. Our goal is to obtain an excellent performance on unseen values. Intuitively, high diversity of slot values make the model more difficult to learn from slot values. Consequently, an interesting question is:

> *If we greatly increase the diversity of slot values in the train set, will the model prefer inferring slot values from slot contexts?*

Besides the diversity of slot values, we also notice that slot values in the training set obey a power-law distribution, a few values occupy more data samples, which may have a negative effect on the generalization performance of the model.

To clarify the impact of distribution and diversity on generalization performance, we contact experiments as follows. First, duplicate the WoZ training set 10 times to accommodate a large set of values. Second, we adjust the diversity of slot values by controlling the number of unique slot values in the training set. We set the numbers of unique values with $[70, 70 \times \alpha^1, \cdots, 70 \times \alpha^{29} = N]$ (*i.e.*, totally 30 numbers), where 70 and $N = 3000$ is the minimum and maximum number of unique slot values respectively. We sample the values in this way to present a clear trend for increasing number of values. For each number, we replace the values in the training set with collected food names to obtain two constructed training sets, obeying power law distribution and uniform distribution respectively. Third, we train the baseline model using the two constructed training sets and test on the synthetic test set.

As shown in Figure 2 (left), the F1-scores of the synthetic test set (*i.e.*, all unseen values) increase rapidly with the increasing number of unique slot values and gradually converge. When the number of unique values is relatively small, the performance of the model is slightly worse when the slot values with the power-law distribution than the uniform distribution. But when there are enough slot values in the training set, the difference between the two distributions can be ignored. Hence, we conclude that the generalization capability of the model is positively correlated to the diversity of values, and the uniform distribution is a better choice when constructing a new training set.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

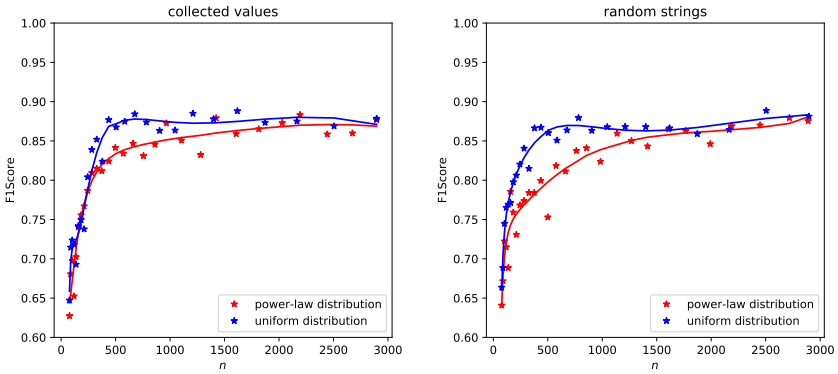

Fig. 2. Performance on T'_{woz} when using the synthetic WoZ training set with different size of value set. In the left part, the datasets are constructed by duplicating the original training set 10 times and replacing slot values with collected food names; in the right part values are replaced with randomly generated strings. The blue lines and points in the figure mean values obey a uniform distribution, and the red lines and points mean values obey a power-law distribution.

4.3 Random strings also work well

We have known that increasing the diversity of slot values can help the copy-mechanism based DST model to obtain a better generalization capability. But in a real-world task-oriented dialogue system, the ontology often contains many slots that have non-enumerable values. As shown in Figure 2, the model needs lots of unique values to produce a satisfactory generalization performance, but collecting so many values for each slot cost a lot. Therefore, we wonder if the randomly generated strings can substitute the real slot values.

We design a simple function `randstr(strlen)` to generate random strings. The function takes string length as input and output a randomly generated string, it samples chars from a fix char sequence, we define the char sequence as 26 lowercase letters and 3 spaces[5]. Spaces are used to generate multi-words values, for example, when `strlen` is 10, the probability of generating multi-words values is about $1 - (26/29)^{10} \approx 0.66$.

We use the same experimental setup as that in Section 4.2, but use random strings generated by `randstr` function to control the diversity of slot values, the results are shown in Figure 2 (right). What can be clearly seen in Figure 2 is that the random strings work well as the collected real slot values. Now we reach a new conclusion: randomly generated strings can be used to enhance the generalization performance of copy-mechanism based DST model.

[5] in this paper we use 3 spaces and `strlen`=10 to generate more natural multi-words values

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

5 Interactive Data Augmentation for DST

We have shown in Section 4 that it is a favorable strategy to employ uniformly distributed random strings to increase the diversity of slot values. In the previous experiments, we duplicate the training set 10 times and set the maximum number of unique slot values to 3000, but we have no idea about whether we have reached the best performance. So how many data do we need to duplicate and how many random values do we need to generate? To answer these questions, we propose a simple data augmentation algorithm and then evaluate it experimentally.

Algorithm 1 Data Augmentation Algorithm

Input: training set U, synthetic dev set D' and test set T', randomly initialized $Tracker$.
Hyperparameters: sampling bag size s_{bag}, training epochs r at each turn and the patience of early stop mechanism p.
Output: the $Tracker$ and its performance on T'

Replace all slot values in U with a string $'[token]'$;
$best\_round$=0, $round$=0, $best\_F1$=0;
while $round - best\_round < p$ **do**
  Sample s_{bag} data samples from U;
  Replace all $'[token]'$ with random strings generated by `randstr`, get U_{bag};
  Train the $Tracker$ on U_{bag} for r epochs;
  Test the $Tracker$ on D', get the F1score $F1$;
  if $F1 > best\_F1$ **then**
    $best\_F1 = F1$;
    $best\_round = round$;
  end if
  $round = round + 1$;
end while
Test the $Tracker$ on T', get the final generalization performance P;
return $Tracker$, P

5.1 Approach

Traditional data augmentation approaches usually work in the data preprocessing phase, while it is hard in our case to determine how many data we need to duplicate and how many random values we need to generate. Thus, we design a data augmentation algorithm that works in the training phase, as shown in Algorithm 1. Firstly, we replace all slot values in the training set U with a unique token $[token]$. Secondly, we sample from the training set repeatedly. At each sampling process, we sample s_{bag} data samples and replace all $[token]$ with different random strings to increase the diversity obeying the uniform distribution; re-calculate the start and end positions of random values to update the labels, so we get a smaller training set U_{bag}. We train the model several epochs on U_{bag}

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

and test it on synthetic develop set D'. The entire training process is controlled by an early stop mechanism, after p rounds sampling that fail to produce a better performance, the algorithm will be terminated and report the final performance on synthetic test set T'.

Our algorithm doesn't need to generate all data before training, so it doesn't need to determine how much data to duplicate and how many values to generate. It could reach the satisfactory results at a low cost.

5.2 Performance

We evaluate our data augmentation algorithm on three datasets: WoZ-sub, DSTC2-sub and Multi-WoZ-sub. We focus on slots that have non-enumerable values (see Table 1) and present experimental results in Table 3.

Dataset	Model	original test set	synthetic test set
WoZ-sub	baseline	0.9241*	0.0820
	+DA	0.9195$_{(\downarrow 0.5\%)}$	0.8819$_{(\uparrow \mathbf{975.5\%})}$
DSTC2-sub	baseline	0.9850*	0.0328
	+DA	0.9605$_{(\downarrow 2.5\%)}$	0.9460$_{(\uparrow \mathbf{2784.1\%})}$
Multi-WoZ-sub	baseline	0.9297*	0.3607
	+DA	0.9032$_{(\downarrow 2.85\%)}$	0.8847$_{(\uparrow \mathbf{145.27\%})}$

Table 3. F1 scores of our data augmentation(DA) approach on three datasets' test sets and synthetic test sets. * means in which all values in test set are visible to the model. Hyper parameters choice are as follows: bag size $s_{bag} = 1600$ for WoZ-sub, $s_{bag} = 3200$ for DSTC2-sub and $s_{bag} = 8000$ for Multi-WoZ-sub, training epochs at each sampling round $r = 10$, the patience of early stop mechanism $p = 5$.

We can observe from Table 3, with our data augmentation algorithm, the performance of the baseline model on unseen values (synthetic test set) improves significantly, and the performance on seen values (original test set) decrease slightly. We use only one slot in the first two datasets, to further demonstrate the effectiveness of our algorithm on a large-scale data set, we test our algorithm on Multi-WoZ-sub dataset that contains 13 slots, results show that it performs well when there are multiple slots.

5.3 Hyperparametric Analysis

Besides the patience p of the early-stop mechanism, there are two key hyper-parameters in the proposed algorithm, the bag size s_{bag} at each sampling round and the epochs r trained on U_{bag}. We analyzed the impact of these two parameters on the algorithm, and the results are shown in Figure 3.

From Figure 3 (left) we can find that given a s_{bag}, a larger r gains a higher performance, but with the s_{bag} increased, the performance gains decreases rapidly. On the other hand, for a certain r, a larger s_{bag} brings a higher performance

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

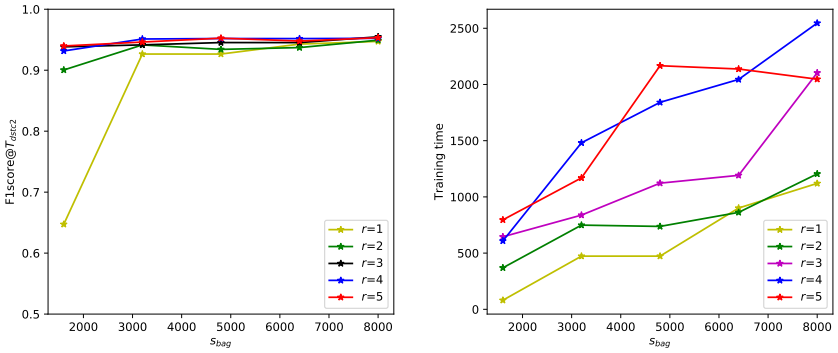

Fig. 3. Performance on T'_{dstc2} of our algorithm. The left part presents the effect of r and s_{bag} on the generalization ability. The right part provides the training time of different r and s_{bag}. We use patience $p = 5$ to produce the results shown in Figure.

but it also decreases rapidly with the increasing r. Our goal is to get the best generalization performance with the minimal cost, Figure 3 (right) presents the training time of different r and s_{bag}, Although the time spent on training increases with both r and s_{bag}, we can conclude that a smaller s_{bag} with a larger r is a suitable choice, which can reach the satisfactory results with lower cost.

6 Conclusion and Future Work

This paper focuses on improving the generalization capability of copy-mechanism based models for DST task, especially for the slots that have non-enumerable values. Our conclusions include: Firstly, the copy-mechanism model is easy to over-fit on visible slot values, especially when there are few values, which is the crucial reason for unsatisfactory generalization performance. Secondly, the model's generalization improves dramatically with the increasing diversity of slot values. Thirdly, data augmentation for copy-mechanism models using random strings is feasible and effective in improving the generalization of these models. The interactive data augmentation approach proposed based on these observations shows its effectiveness on three widely used datasets.

In future work, we can study the effect of character-level features of the slot values since the values for the same slot often share somehow similar spelling. In addition, the effect of the diversity of contexts remains unexplored, which may help reduce computational cost further.

7 Acknowledgements

This research is supported in part by the National Key Research and Development Program of China under Grant 2018YFC0806900 and 2017YFB1010000.

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

References

1. Budzianowski, P., Wen, T.H., Tseng, B.H., Casanueva, I., Ultes, S., Ramadan, O., Gašić, M.: Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. arXiv preprint arXiv:1810.00278 (2018)
2. Chen, D., Fisch, A., Weston, J., Bordes, A.: Reading wikipedia to answer open-domain questions. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1870–1879 (2017)
3. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805 (2018)
4. Gao, S., Sethi, A., Agarwal, S., Chung, T., Hakkani-Tur, D.: Dialog state tracking: A neural reading comprehension approach. In: Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue. pp. 264–273 (2019)
5. Henderson, M., Thomson, B., Williams, J.D.: The second dialog state tracking challenge. In: Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL). pp. 263–272 (2014)
6. Henderson, M., Thomson, B., Williams, J.D.: The third dialog state tracking challenge. In: 2014 IEEE Spoken Language Technology Workshop (SLT). pp. 324–329. IEEE (2014)
7. Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural computation **9**(8), 1735–1780 (1997)
8. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014)
9. Mrkšić, N., Séaghdha, D.O., Wen, T.H., Thomson, B., Young, S.: Neural belief tracker: Data-driven dialogue state tracking. arXiv preprint arXiv:1606.03777 (2016)
10. Ramadan, O., Budzianowski, P., Gašić, M.: Large-scale multi-domain belief tracking with knowledge sharing. arXiv preprint arXiv:1807.06517 (2018)
11. Rastogi, A., Hakkani-Tür, D., Heck, L.: Scalable multi-domain dialogue state tracking. In: 2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). pp. 561–568. IEEE (2017)
12. Ren, L., Ni, J., McAuley, J.: Scalable and accurate dialogue state tracking via hierarchical sequence generation. In: Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP). pp. 1876–1885. Association for Computational Linguistics, Hong Kong, China (Nov 2019). https://doi.org/10.18653/v1/D19-1196, `https://www.aclweb.org/anthology/D19-1196`
13. Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., Salakhutdinov, R.: Dropout: A simple way to prevent neural networks from overfitting. Journal of Machine Learning Research **15**, 1929–1958 (2014), `http://jmlr.org/papers/v15/srivastava14a.html`
14. Vinyals, O., Fortunato, M., Jaitly, N.: Pointer networks. In: Advances in Neural Information Processing Systems. pp. 2692–2700 (2015)
15. Wen, T.H., Vandyke, D., Mrksic, N., Gasic, M., Rojas-Barahona, L.M., Su, P.H., Ultes, S., Young, S.: A network-based end-to-end trainable task-oriented dialogue system. arXiv preprint arXiv:1604.04562 (2016)
16. Williams, J., Raux, A., Ramachandran, D., Black, A.: The dialog state tracking challenge. In: Proceedings of the SIGDIAL 2013 Conference. pp. 404–413 (2013)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59

17. Wu, C.S., Madotto, A., Hosseini-Asl, E., Xiong, C., Socher, R., Fung, P.: Transferable multi-domain state generator for task-oriented dialogue systems. In: Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). Association for Computational Linguistics (2019)
18. Xu, P., Hu, Q.: An end-to-end approach for handling unknown slot values in dialogue state tracking. arXiv preprint arXiv:1805.01555 (2018)
19. Young, S., Gašić, M., Thomson, B., Williams, J.D.: Pomdp-based statistical spoken dialog systems: A review. Proceedings of the IEEE **101**(5), 1160–1179 (2013)
20. Zhong, V., Xiong, C., Socher, R.: Global-locally self-attentive encoder for dialogue state tracking. In: Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers). pp. 1458–1467 (2018)

ICCS Camera Ready Version 2021
To cite this paper please use the final published version:
DOI: 10.1007/978-3-030-77961-0_59