

PROMPT POOL BASED CLASS-INCREMENTAL CONTINUAL LEARNING FOR DIALOG STATE TRACKING

Hong Liu^{1,3,†}, Yucheng Cai^{1,3,†}, Yuan Zhou^{1,3,†}, Zhijian Ou^{*,1,3}, Yi Huang^{2,3}, Junlan Feng^{2,3}

¹Speech Processing and Machine Intelligence (SPMI) Lab, Tsinghua University, Beijing, China

²China Mobile Research Institute, Beijing, China

³Tsinghua University-China Mobile Communications Group Co., Ltd. Joint Institute, Beijing, China

ABSTRACT

Continual learning is crucial for dialog state tracking (DST) in dialog systems, since requirements from users for new functionalities are often encountered. However, most of existing continual learning methods for DST require task identities during testing, which is a severe limit in real-world applications. In this paper, we aim to address continual learning of DST in the class-incremental scenario (namely the task identity is unknown in testing). Inspired by the recently emerging prompt tuning method that performs well on dialog systems, we propose to use the prompt pool method, where we maintain a pool of key-value paired prompts and select prompts from the pool according to the distance between the dialog history and the prompt keys. The proposed method can automatically identify tasks and select appropriate prompts during testing. We conduct experiments on Schema-Guided Dialog dataset (SGD) and another dataset collected from a real-world dialog application. Experiment results show that the prompt pool method achieves much higher joint goal accuracy than the baseline. After combining with a rehearsal buffer, the model performance can be further improved.

Index Terms— Dialog state tracking, continual learning, prompt pool

1. INTRODUCTION

Task-oriented dialog (TOD) systems are designed to help users accomplish specific goals such as booking flights and finding restaurants. Dialog state tracking (DST) is an important component in TOD systems, which tracks user goals by inferring structured dialog states expressed in terms of slots and values, as shown in Figure 1 [1]. The methods for building DST have been gradually advancing from classification-based [2, 3] to sequence-to-sequence generation based [4, 5, 6, 7, 8]. Current DST models are mostly trained in an offline manner, assuming that the domains and required functionalities are fixed through time and that all

[†] Equal contribution.

^{*} Corresponding author (ozj@tsinghua.edu.cn).

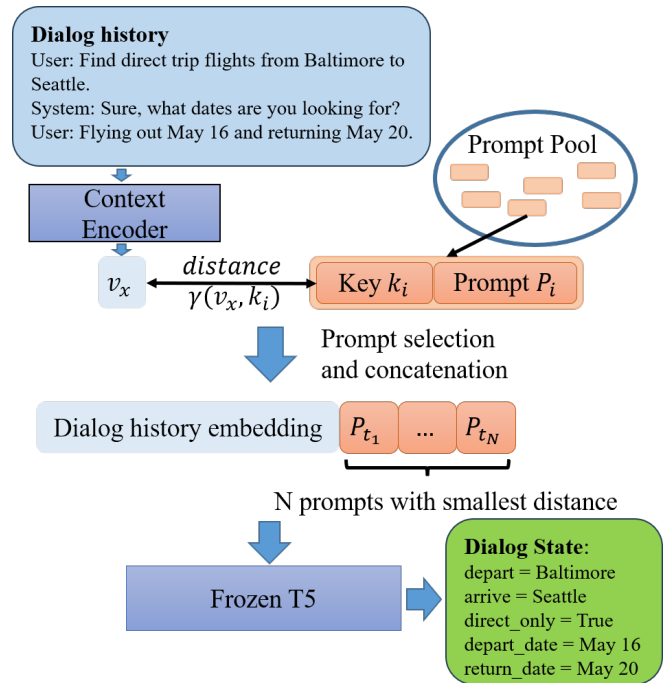


Fig. 1. An overview of prompt pool based continual training for DST. Prompt vectors that are close to the context vector will be selected from the pool and concatenated with the dialog history embeddings. The T5 model then takes all the embeddings as input and outputs the dialog state.

training data can be accessed beforehand. However, a practical DST model often has to face new tasks. A common requirement is to add new domains for dialogs. It is costly if the DST model is re-trained when each new task is added. Therefore, continual learning (CL), which refers to expanding a model to new tasks efficiently without forgetting old tasks, i.e., catastrophic forgetting [9], is crucial for TOD systems.

To overcome catastrophic forgetting, three main classes of continual learning algorithms have been developed: *rehearsal*, which uses an replay buffer to recall previous learned task [10, 11, 12], *regularization*, which adds regularization term to the loss to avoid forgetting [13, 14], and *architec-*

tural, which trains task-specific component for each task [15, 16, 17, 18, 19]. However, rehearsal-based methods tend to exhibit decreased performance when the buffer size is reduced, and they are unsuitable for scenarios where the data privacy is a concern. Regularization-based methods partially mitigate catastrophic forgetting without the need to store past examples, but they fail to achieve desirable performance in demanding scenarios or intricate datasets [11]. Architecture-based methods aim to have dedicated components for each task and are more flexible and efficient than the above two classes of methods. Those task-specific components can be achieved by various approaches such as training a separate adapter [17] or applying network pruning [16] for each task. However, most architecture-based methods require that the task identity is known during testing. This presents a severe limitation for their application in class-incremental continual learning, for which task identification is necessary at test time. We leave a short introduction to the three basic scenarios of continual learning [20], i.e., task-incremental, domain-incremental and class-incremental, to the section of related work.

In this paper, we are interested in continual learning of DST in the class-incremental learning scenario (namely the task identity is unknown in testing), which is mostly under-explored. The training data for different domains arrives in a sequence, thus constituting a sequence of tasks. The DST model needs to be incrementally trained and finally perform well under all tasks. A recent work in [18] applied continual prompt tuning (CPT) to DST, where it fixes the pretrained language model, trains task-specific prompt vectors for each task, and concatenates those prompts with the context embeddings as the final input embeddings. CPT achieved impressive performance in continual learning of DST. However, CPT cannot work in the class-incremental scenario, because it needs to know the corresponding prompts and slot definitions for the current task before generating dialog states.

Inspired by the learning to prompt (L2P) method for image classification [19], we propose a prompt pool based continual learning of DST, which can fully support the class-incremental scenario. Specifically, we maintain a prompt pool which contains a set of prompt vectors. In addition, each prompt is associated with a key vector for selecting prompts. For a dialog turn from an arbitrary task, we select prompts from the key-value paired prompt pool according to the distance between the context vector and key vectors. The concatenation of the dialog history embeddings and the selected prompts will be sent into a pretrained model with encoder-decoder structure such as T5 [21] to predict the dialog state. The parameters of all the prompts and keys will be updated during training, while the context encoder and the pretrained encoder-decoder model are frozen. The overview of our continual learning framework can be seen in Figure 1.

We conduct experiments of DST on the widely-used Schema-Guided Dialog dataset (SGD) [22] and another

Chinese dataset collected from a real-world dialog application. We model DST as a sequence-to-sequence generation problem and adjust the sequence format to fit in the class-incremental scenario. The results show that the prompt pool method achieves much higher joint goal accuracy than baseline AdapterCL [17] in class-incremental setting. Moreover, we combine prompt pool with a rehearsal buffer and modify the selection objective for keys, which further improved the model performance.¹

2. RELATED WORK

2.1. Continual Learning

According to training methods, three main classes of continual learning algorithms have been developed: rehearsal-based, regularization-based and architecture-based. Rehearsal-based methods employ a data buffer to store samples from previous tasks, which are then used for training along with the data from the current task [10, 11, 12]. Regularization-based methods restrict the plasticity of the model by constraining the learning rate at important parameters for previous tasks [13, 14]. Architecture-based methods aim to train dedicated components for each task. These task-specific components can be achieved by dynamic expanding network structure [15], iteratively applying network pruning and modification [16], training a separate adapter [17] or training task-specific prompts [18, 19] for each task.

According to test scenarios, continual learning can be divided into task-incremental, domain-incremental, and class-incremental [20]. Task-incremental learning is the simplest scenario, where the model is aware of the task identity of the current data during testing. Domain-incremental learning is more challenging than task-incremental learning, since the model lacks information about the task identity during testing, but the data labels remain consistent across all tasks, e.g., all tasks are binary classification task. Class-incremental learning is the most complex category among these three, but it is also the closest to real-world scenarios. In class-incremental learning, the model is unaware of the task identity and the labels differ across different tasks.

2.2. Prompt-based Natural Language Processing

Recent studies have found that using a textual prompt can better align pretrained language models to downstream tasks [23, 24]. Prompt engineering either manually designs prompts [25] or generates prompts automatically [26, 27]. Different from prompt engineering, prompt-tuning adds new tunable prompt tokens to the input, while keep the pretrained model frozen. The added prompts serve as context and affect all following transformer layers, and their embeddings are learned through back-propagation. It is shown that prompt-tuning is parameter-efficient and becomes more competitive with fine-tuning as the model size grows [28]. Prompt-tuning is competitive for continual learning, as only the soft prompts are

¹The code is released at <https://github.com/thu-spmi/PPT2DST>

tuned, instead of the whole pretrained language model. Our work proposes to use the prompt pool to leverage the advantage of prompt-tuning, while letting the model to identify tasks and selecting the most appropriate prompts automatically to deal with the class-incremental continual learning problem.

2.3. Continual Learning in TOD Systems

Continual learning has been studied in building TOD systems. In [29], a regularization-based method, adaptive elastic weight consolidation (AEWC), is utilized to complete continual learning in DST and dialog management. In [17], separate adapters are trained during continual learning and applied to natural language understanding (NLU), DST and natural language generation (NLG). In [30], rehearsal-based and regularization-based methods are combined for NLG in TOD systems. In [16], continual learning of NLG is performed by iterative pruning, expanding and masking the network. A recent work in [18] achieve continual learning of DST by applying prompt-tuning [31], where the pre-trained language model is fixed and the added prompts tokens are tuned to adapt the language model to a sequence of tasks. However, most of previous studies concentrate on the task-incremental or domain-incremental scenarios, which limit those methods in practical applications. In this work, we aim to address the most challenging class-incremental learning of DST.

A relevant prior study to our work is AdapterCL [17], which assumes the task identity is unknown during testing and select the adapter according to the perplexity. However, in [18], AdapterCL is found to be not parameter-efficient enough, where AdapterCL needs 20 times parameters to catch up with the performance of continual prompt tuning. Though the prompt tuning method has shown its effectiveness in continual learning of DST [18], it is not suitable for the class-incremental scenario of TOD system. Our work is inspired by [19], which propose the L2P (also known as prompt pool) method for image classification. The method maintains a prompt pool for continual learning and selects the prompts using key matching. In this work, we further extend the prompt pool method to the class-incremental learning of DST.

3. METHOD

3.1. Continual Learning

In the class-incremental scenario, there are a sequence of tasks $\mathcal{T} = \{\mathcal{T}_1, \dots, \mathcal{T}_T\}$. The training data for different tasks arrives in a sequence. The model needs to be incrementally trained and finally perform well under all tasks. Denote the corresponding data by $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_T\}$, where \mathcal{D}_t denotes the data for the task \mathcal{T}_t . \mathcal{D}_t contains multiple data samples (x_t^i, y_t^i) where $x_t^i \in \mathcal{X}_t$ and $y_t^i \in \mathcal{Y}_t$ denote the i -th input sample and label. We will omit the index t and i in x_t^i later for simplicity, and add a subscript to denote a particular sample such as in Eq. (1) and (2) for a particular dialog turn k .

3.2. Dialog State Tracking

For dialog state tracking (DST), we need to use the dialog history to predict the dialog state for each turn. The dialog state is a set of slot value pairs: $\{(s_1, v_1), \dots, (s_{n_t}, v_{n_t})\}$, where s and v denote slot and value respectively, and n_t is the number of slots used in the task \mathcal{T}_t . Usually, all the slots are predefined and DST is to predict the values. Task-oriented dialogs often consist of dialogs from different domains, such as restaurant service or hotel service. For continual learning of DST, the domain identity is treated as task identity, which is unknown in testing in the class-incremental setting.

Generally, we formulate DST as text-to-text generation. The DST model accepts an input token sequence and outputs the dialog state, which is also represented by a token sequence. For the input sequence, [18] concatenates slot descriptions and sentinel tokens after dialog history to better predict the value of each slot. However, the task identity is unknown during testing in class-incremental setting, which means that the model does not know which slots are involved in current turn. Therefore, we only take the dialog history as the input sequence, that is

$$x_k = u_1 \oplus r_1 \oplus \dots \oplus u_{k-1} \oplus r_{k-1} \oplus u_k \quad (1)$$

where u_k, r_k denote the user utterance and system response in the k -th dialog turn, and \oplus denote the concatenation operation. For the output sequence, not only the slot values but also the task identity need to be predicted. To simplify the output format, we preset the order of slots in output sequence, so that we only need to predict the values in a specific order during testing. We use special tokens $[s_0], \dots, [s_{n_t}]$ to separate values. The empty values are set to be 'none' in the sequence. The output sequence can be formulated as:

$$y_k = [s_0] id_t [s_1] v_1^k, \dots, [s_{n_t}] v_{n_t}^k \quad (2)$$

where id_t is the identity (i.e., task name) of task \mathcal{T}_t . For simplicity, we omit the turn index k in subsequent formulas.

3.3. Prompt Pool

3.3.1. Prompt Tuning

Prompt tuning [31] simply conditions frozen T5-like language models [18] to perform down-stream NLP tasks by learning prompt parameters that are concatenated to the input tokens to instruct the model prediction. However, ordinary prompt tuning is not applicable to class-incremental learning where the task identity is unknown in testing. [19] proposed to learn a prompt pool memory space which is served as parameterized instructions for the pretrained model to learn tasks sequentially. The prompt pool is defined as:

$$\mathbf{P} = \{P_1, P_2, \dots, P_J\} \quad (3)$$

where $P_j \in \mathbb{R}^{L_p \times D}$ is a single prompt with token length L_p and embedding size D , and J is the number of prompts in the pool. For each task, N prompts, $P_{i_1}, P_{i_2}, \dots, P_{i_N}$, will

be selected from the prompt pool and concatenated after the embeddings of the dialog history. Let $E(x) \in \mathbb{R}^{|x| \times D}$ denote the embeddings of the input sequence x with token length $|x|$. Then the vector sequence $E_p(x)$ fed into the pretrained model like T5, denoted by $f_\theta(\cdot)$, can be formulated as:

$$E_p(x) = E(x) \oplus P_{i_1} \oplus \dots \oplus P_{i_N} \quad (4)$$

3.3.2. Prompt Selection

To select prompts from the prompt pool, [19] model each prompt as a key-value pair: $\{(k_1, P_1), \dots, (k_J, P_J)\}$. $k_i \in \mathbb{R}^{D_K}$ is the key for the i -th prompt and we denote the set of all keys by \mathbf{K} . Intuitively, we can select prompts according to the distance between x and k_i . Specifically, the input sequence will be sent into a pretrained encoder model q_ϕ to obtain the context vector $c_x \in \mathbb{R}^{D_K}$, for example, the output vector of BERT model corresponding to the '[CLS]' token. Then the distances between c_x and all the keys will be calculated and N keys with the smallest distances will be selected. The dialog history embeddings and the prompts corresponding to the selected N keys will be concatenated as in Eq. (4). During training, to ensure that each key in the pool can be selected, we follow [19] to directly select $k_{Nt:N(t+1)-1}$ for the task \mathcal{T}_t , which is motivated by diversifying prompt selection in [19].

3.3.3. Optimization of Prompt Pool

The optimization of the prompt pool can be divided into two parts. The first part is the cross entropy loss between the model output and the label. The second part is the loss between the context vector and the selected prompt keys.

$$\mathcal{L} = CE(f_\theta(E_p(x)), y) + \lambda \sum_{k_i \in \mathbf{K}_x} \gamma(c_x, k_i) \quad (5)$$

where CE denotes the cross entropy loss which is averaged over all tokens in the output sequence, γ is a function that calculates Euclid distance and feeds it into a sigmoid function, λ is the weight of the second part loss, and \mathbf{K}_x is the set of keys selected from the pool for the input sequence x . The whole training algorithm can be seen in Algorithm 1.

3.3.4. Rehearsal Buffer

Utilizing rehearsal buffer can improve model performance effectively when past data are accessible. For the task \mathcal{T}_t , the rehearsal buffer is composed of a fixed number of dialogs selected from previous $t - 1$ tasks, which we denote as $M_{<t}$. The new dataset for the task \mathcal{T}_t is $\mathcal{D}_t \cup M_{<t}$. It is worth noting that the second term in Eq. (5) is not applicable to the rehearsal-based method, because it will shorten the distance between the context vectors from $M_{<t}$ and keys from \mathcal{T}_t . To address this issue, we change the second term in Eq. (5) to a binary cross entropy loss. The loss function of the task \mathcal{T}_t can be written as

$$\mathcal{L} = CE(f_\theta(E_p(x)), y) + \lambda \sum_{k_i \in \mathbf{K}_x} BCE(\gamma(c_x, k_i), \mathbf{I}(x \in \mathcal{D}_t)) \quad (6)$$

Algorithm 1 Prompt Pool Training (PPT) for DST

Require: Frozen pretrained model f_θ , frozen encoder q_ϕ , task number T , a sequence of data \mathcal{D} , prompt pool \mathbf{P} , prompt keys \mathbf{K} , prompt number N for each task, batch size B , epochs E ;

Randomly initialize \mathbf{P}, \mathbf{K} ;

for $t=1$ to T **do**

Select N keys and prompts $\mathbf{K}_x = \{K_{Nt}, \dots, K_{N(t+1)-1}\}$, $\mathbf{P}_x = \{P_{Nt}, \dots, P_{N(t+1)-1}\}$;

for $e=1$ to E **do**

Obtain a mini-batch of data $\{(x_b, y_b)\}_{r=b}^B$;

Calculate the context vector $c_{x_b} = q_\phi(x_b)$ for all input samples;

Concatenate the embedding of the input sequence with the selected prompts and obtain an embedding batch $\{(E_p(x_b), y_b)\}_{r=b}^B$;

Calculate the loss $\mathcal{L} =$

$$\frac{1}{B} \sum_{b=1}^B [CE(f_\theta(E_p(x_b)), y_b) + \lambda \sum_{k_i \in \mathbf{K}_x} \gamma(c_{x_b}, k_i)];$$

end for

Update the selected keys \mathbf{K}_x and prompts \mathbf{P}_x using the loss \mathcal{L} ;

end for

where $\mathbf{I}(x \in \mathcal{D}_t)$ is an indicator function and BCE is the binary cross entropy loss function, where $BCE(x, y) = -y \log x - (1 - y) \log(1 - x)$. The algorithm with rehearsal buffer is shown in Algorithm 2 in Appendix.

4. EXPERIMENTS

4.1. Dataset

We conduct our experiments on Schema-Guided Dialog dataset (SGD) [22] that has 44 services over 19 domains. Like [18], we treat each service as a task and only consider dialogs involving a single service. We randomly select 15 tasks and split the dialogs of one service into train/val/test sets with the ratio of 7:1:2. More details about data statistics can be found in Table 6 in Appendix. To examine the performance of the method in real-world applications, we conduct experiments on the China Mobile Pickup dataset (CM-Pickup), collected from a real-world dialog application. The purpose is to automatically pickup the incoming call when the phone owner is not available, via dialog state tracking and dialog management. CM-Pickup has 39 domains and we retain 16 domains that have more than 100 dialog sessions and discard other domains.

4.2. Metrics

Generally, we evaluate the DST performance using Joint Goal Accuracy (JGA), which calculates the proportion of dialog turns that all the slot values are correctly predicted. Denote $a_{j,i}$ as the JGA on the test set of the i -th task right after training on the j -th task. A normal measure of the performance of

continual learning is the average JGA on all tasks after training on the final task, that is:

$$JGA_{avg} = \frac{1}{T} \sum_{t=1}^T a_{T,t} \quad (7)$$

Besides, we use $f_{t,i}$ to represent the forgetting index of the i -th task after training on the task \mathcal{T}_t .

$$f_{t,i} = \max_{j \in [i,t]} a_{j,i} - a_{t,i} \quad (8)$$

We also calculate the accuracy of key selection during testing, which is denoted as Acc_{key} . Note that one task corresponds to multiple keys, so counting as correct means that all the keys are selected correctly.

4.3. Baseline

We abbreviate the prompt pool training method as PPT and compare it with another class-incremental baseline AdapterCL [17], which trains a residual adapter for each task and select the adapter with lowest perplexity during testing. For the sake of fairness, we adjust the parameter size of AdapterCL to be close to that of prompt pool. To improve the performance, we equip PPT with a rehearsal buffer (PPT-R), where we randomly select 50 samples from each task as memory.

We train models using multitask prompt tuning (MPT) method and oracle continual prompt tuning (OCPT) method as the *upper-bound*. The first method trains N prompts using all tasks’ data simultaneously. The second method trains N prompts for each task sequentially but the task identity is provided during testing (hence called oracle). Both are trained using only the first part loss in Eq. (5).

5. RESULTS

5.1. Main Results

Table 1 shows the average JGA of the model on 15 tasks after continual learning over SGD. The main findings are as follows: 1) PPT achieves higher JGA_{avg} than AdapterCL; 2) The JGA_{avg} of PPT and PPT-R are still lower than OCPT, which provides task identities in testing. This illustrates the challenge of class-incremental learning; 3) Both PPT and PPT-R have a high accuracy of key selection during testing, indicating that the prompt pool method can be well applied to class-incremental learning scenarios in DST; 4) The rehearsal buffer can improve JGA_{avg} and Acc_{key} effectively.

To better demonstrate how the model performance varies during continual learning, we calculate the forgetting index during training. We only show the first 6 tasks in Table 2 due to space limitations. Interestingly, the forgetting index often increases after training on similar tasks. For example, the forgetting index of the second task, flights.1 increases to 0.176 after training on flights.3. We speculate that this is because the model cannot distinguish between two similar tasks well (The data distributions of them is close to each other), leading to errors in predicting the task identity of previous tasks.

Method	JGA_{avg}	Acc_{key}
OCPT	0.481	-
MPT	0.614	-
AdapterCL	0.306	-
PPT	0.346	0.783
PPT-R	0.363	0.811

Table 1. Average joint goal accuracy on 15 tasks over SGD. The first block contains the two methods that represent the upper bound and the second shows class-incremental results of different methods. We also show the key selection accuracy for the PPT methods.

Task name	Forgetting index
services.4	[0.000, 0.000, 0.000, 0.000, 0.000, 0.000]
flights.1	[0.000, 0.000, 0.000, 0.000, 0.000, 0.000]
services.3	[0.115 , 0.000, 0.000, 0.000, 0.000, 0.000]
flights.3	[0.115, 0.176 , 0.000, 0.000, 0.000, 0.000]
trains.1	[0.115, 0.176, 0.000, 0.000, 0.000, 0.000]
homes.2	[0.115, 0.176, 0.000, 0.000, 0.000, 0.000]

Table 2. The forgetting indices of the first 6 tasks over SGD. Each row contains 6 indices, corresponding to the forgetting index of the current task for the first 6 tasks.

Table 3 show JGA and Acc_{key} of PPT and PPT-R on different tasks after continual learning. It can be found that JGA and Acc_{key} of most tasks improved after adding a rehearsal buffer. Nonetheless, JGA and Acc_{key} of a few tasks such as rentalcars.3 have significantly decreased with a rehearsal buffer. After analysis, we found that after adding a rehearsal buffer, the model has a probability of over 70% misjudging rentalcars.3 as rentalcars.1. This phenomenon is consistent with the speculation above. In fact, these similar tasks should be merged into one in a more ideal continuous learning scenario.

For another dataset CM-Pickup, we compare PPT-R with OCPT. The JGA of each task is shown in Figure 2. The overall results are similar to Table 1, where PPT-R achieves slightly lower JGA than the upper-bound OCPT.

5.2. Ablation Study

To understand the similarities between different tasks in SGD, we utilize the t-SNE algorithm [32] to perform dimension reduction on all the encoded context vectors. The results are shown in Figure 3. It can be seen that the number of clusters in the figure is less than the number of tasks, 15. For instance, in the middle of the figure, the points of flights.1 and flights.3 are closely intertwined. This indicates that some tasks are similar to each other, which increases the difficulty for the model to distinguish between different tasks.

To reveal the role of the modified loss function, we conduct two ablation experiments on the basis of PPT-R, and the results are shown in Table 4. The first experiment (PPT-R_{prompt.only}) directly removes the second term in Eq. (6), that is, the pool of keys, \mathbf{K} , is not updated during training.

Task name	PPT		PPT-R	
	JGA	Acc_{key}	JGA	Acc_{key}
services_4	0.510	0.793	0.538	0.981
flights_1	0.537	0.741	0.563	0.802
services_3	0.534	1.000	0.555	0.990
flights_3	0.422	0.922	0.353	0.836
trains_1	0.368	0.983	0.419	1.000
homes_2	0.144	0.311	0.230	0.547
rentalcars_2	0.086	0.459	0.416	0.989
restaurants_1	0.497	1.000	0.473	1.000
music_1	0.324	1.000	0.211	0.951
hotels_4	0.014	0.014	0.255	0.355
media_2	0.254	1.000	0.338	0.972
hotels_3	0.264	0.808	0.228	0.829
rentalcars_3	0.263	0.828	0.020	0.263
hotels_1	0.352	0.880	0.256	0.720
homes_1	0.628	1.000	0.589	0.930
avg	0.346	0.783	0.363	0.811

Table 3. The JGA and Acc_{key} of PPT and PPT-R on 15 tasks after continual learning over SGD.

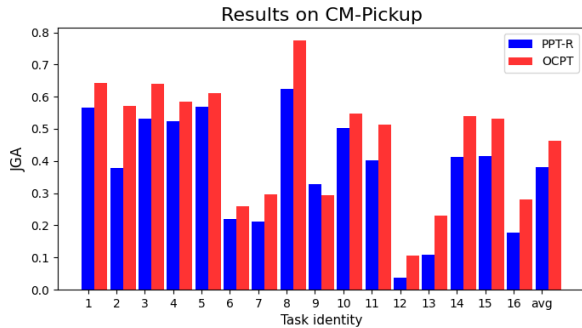


Fig. 2. The joint goal accuracy of PPT-R and OCPT (upper-bound) on each task of CM-Pickup.

The second (PPT-R_{ordinary}) calculates the loss according to Eq. (5) instead of Eq. (6) when utilizing rehearsal buffer. Both PPT-R_{prompt_only} and PPT-R_{ordinary} achieve lower JGA_{avg} and Acc_{key} than PPT-R, and the results of PPT-R_{ordinary} is even lower than those of PPT-R_{prompt_only}. This indicates that the key selection loss in the ordinary loss function in Eq. (5) is unfavorable for prompt pool training with a rehearsal buffer, and the modified loss in Eq. (6) can improve the model performance effectively.

To demonstrate that our methods can scale well with the parameters of the backbone, we conduct experiments with different backbones and report the results in Table 5. The results clearly show that our PPT methods scale well with the backbone size. Using a larger model like T5-base and T5-large continually improve the performance of the JGA_{avg} and Acc_{key} metrics. This finding shows that our PPT methods can potentially work well with large language models, which has become prevalent for NLP tasks recently.

6. CONCLUSION

We propose to address the class-incremental learning problem of DST using the prompt pool method. We maintain a

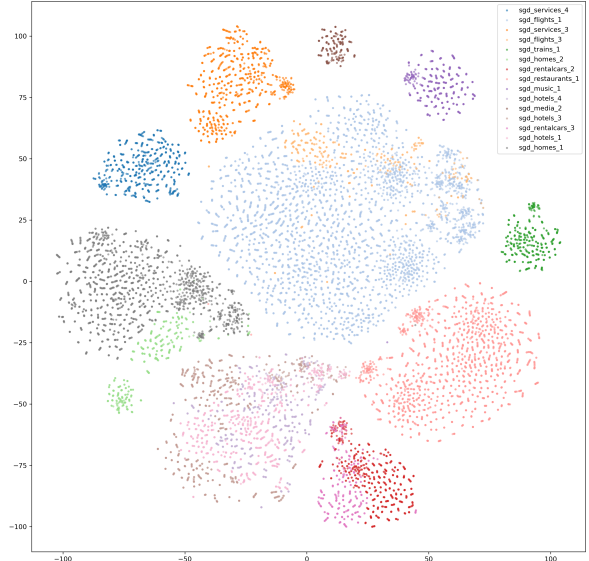


Fig. 3. The distribution of context vectors from different tasks in SGD after t-SNE dimension reduction.

Method	JGA_{avg}	Acc_{key}
PPT-R	0.363	0.811
PPT-R _{prompt_only}	0.336	0.766
PPT-R _{ordinary}	0.334	0.732

Table 4. Ablation study of the modified key selection loss over SGD.

Model	JGA_{avg}	Acc_{key}
T5-small (60M)	0.346	0.783
T5-base (200M)	0.420	0.800
T5-large (750M)	0.464	0.822

Table 5. Ablation study of the scaling effect of the backbone used for prompt tuning over SGD. PPT is used for all different backbones.

prompt pool and select the prompts that are close to the input sequence vector during continual learning. The embeddings of the input sequence and the selected prompts will be concatenated together and sent to a pretrained model to predict the dialog state. We conduct experiments on SGD and CM-Pickup and the results show that the prompt pool method outperforms the baseline. We also combine prompt pool with a rehearsal buffer, which further improves the joint goal accuracy. We hope that this work is helpful for building more flexible generative dialog systems for real-world applications.

7. REFERENCES

- [1] Jason D Williams, Antoine Raux, and Matthew Henderson, “The dialog state tracking challenge series: A review,” *Dialogue & Discourse*, vol. 7, no. 3, pp. 4–33, 2016.
- [2] Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young, “Neural belief tracker: Data-driven dialogue state tracking,” in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2017.
- [3] Yinpei Dai, Zhijian Ou, Dawei Ren, and Pengfei Yu, “Tracking of enriched dialog states for flexible conversational information access,” in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, 2018, pp. 6139–6143.
- [4] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young, “A network-based end-to-end trainable task-oriented dialogue system,” in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*, 2017.
- [5] Bing Liu and Ian Lane, “An end-to-end trainable neural network model with belief tracking for task-oriented dialog,” *Proc. Interspeech 2017*, pp. 2506–2510, 2017.
- [6] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin, “Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures,” in *56th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2018.
- [7] Yichi Zhang, Zhijian Ou, and Zhou Yu, “Task-oriented dialog systems that consider multiple appropriate responses under the same context,” in *The Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.
- [8] Hong Liu, Yucheng Cai, Zhijian Ou, Yi Huang, and Junlan Feng, “Building markovian generative architectures over pretrained lm backbones for efficient task-oriented dialog systems,” in *2022 IEEE Spoken Language Technology Workshop (SLT)*. IEEE, 2023, pp. 382–389.
- [9] Michael McCloskey and Neal J. Cohen, “Catastrophic interference in connectionist networks: The sequential learning problem,” vol. 24 of *Psychology of Learning and Motivation*, pp. 109–165. 1989.
- [10] Cyprien de Masson D’Autume, Sebastian Ruder, Lingpeng Kong, and Dani Yogatama, “Episodic memory in lifelong language learning,” *Advances in Neural Information Processing Systems (NeurIPS)*, vol. 32, 2019.
- [11] Sylvestre-Alvise Rebuffi, Alexander Kolesnikov, Georg Sperl, and Christoph H. Lampert, “iCaRL: Incremental classifier and representation learning,” in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, July 2017.
- [12] David Lopez-Paz and Marc’ Aurelio Ranzato, “Gradient episodic memory for continual learning,” in *Advances in Neural Information Processing Systems*, 2017, vol. 30.
- [13] James Kirkpatrick, Razvan Pascanu, Neil Rabinowitz, Joel Veness, Guillaume Desjardins, Andrei A Rusu, Kieran Milan, John Quan, Tiago Ramalho, Agnieszka Grabska-Barwinska, et al., “Overcoming catastrophic forgetting in neural networks,” *Proceedings of the national academy of sciences*, vol. 114, no. 13, pp. 3521–3526, 2017.
- [14] Zhizhong Li and Derek Hoiem, “Learning without forgetting,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 40, no. 12, pp. 2935–2947, 2018.
- [15] Andrei A Rusu, Neil C Rabinowitz, Guillaume Desjardins, Hubert Soyer, James Kirkpatrick, Koray Kavukcuoglu, Razvan Pascanu, and Raia Hadsell, “Progressive neural networks,” *arXiv preprint arXiv:1606.04671*, 2016.
- [16] Binzong Geng, Fajie Yuan, Qiancheng Xu, Ying Shen, Ruifeng Xu, and Min Yang, “Continual learning for task-oriented dialogue system with iterative network pruning, expanding and masking,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, Aug. 2021, pp. 517–523.
- [17] Andrea Madotto, Zhaojiang Lin, Zhenpeng Zhou, Seungwhan Moon, Paul Crook, Bing Liu, Zhou Yu, Eunjoon Cho, Pascale Fung, and Zhiguang Wang, “Continual learning in task-oriented dialogue systems,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Nov. 2021, pp. 7452–7467.
- [18] Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang, “Continual prompt tuning for dialog state tracking,” in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, May 2022, pp. 1124–1137.
- [19] Zifeng Wang, Zizhao Zhang, Chen-Yu Lee, Han Zhang, Ruoxi Sun, Xiaoqi Ren, Guolong Su, Vincent Perot, Jennifer Dy, and Tomas Pfister, “Learning to prompt for continual learning,” in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022, pp. 139–149.

- [20] Gido M Van de Ven and Andreas S Tolias, “Three scenarios for continual learning,” *arXiv preprint arXiv:1904.07734*, 2019.
- [21] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.
- [22] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan, “Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset,” in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2020, vol. 34, pp. 8689–8696.
- [23] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei, “Language models are few-shot learners,” in *Advances in Neural Information Processing Systems*, H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, Eds., 2020, vol. 33, pp. 1877–1901.
- [24] Timo Schick and Hinrich Schütze, “Exploiting cloze-questions for few-shot text classification and natural language inference,” in *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, 2021, pp. 255–269.
- [25] Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller, “Language models as knowledge bases?,” 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)*, Nov. 2019, pp. 2463–2473.
- [26] Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh, “Autoprompt: Eliciting knowledge from language models with automatically generated prompts,” in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 4222–4235.
- [27] Tianyu Gao, Adam Fisch, and Danqi Chen, “Making pre-trained language models better few-shot learners,” in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, 2021, pp. 3816–3830.
- [28] Brian Lester, Rami Al-Rfou, and Noah Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Nov. 2021, pp. 3045–3059.
- [29] Sungjin Lee, “Toward continual learning for conversational agents,” *arXiv preprint arXiv:1712.09943*, 2017.
- [30] Fei Mi, Liangwei Chen, Mengjie Zhao, Minlie Huang, and Boi Faltings, “Continual learning for natural language generation in task-oriented dialog systems,” in *Findings of the Association for Computational Linguistics: EMNLP 2020*, Nov. 2020, pp. 3461–3474.
- [31] Brian Lester, Rami Al-Rfou, and Noah Constant, “The power of scale for parameter-efficient prompt tuning,” in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, Nov. 2021, pp. 3045–3059.
- [32] Laurens Van der Maaten and Geoffrey Hinton, “Visualizing data using t-sne,” *Journal of machine learning research*, vol. 9, no. 11, 2008.
- [33] Nils Reimers and Iryna Gurevych, “Sentence-BERT: Sentence embeddings using Siamese BERT-networks,” 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)*, Nov. 2019, pp. 3982–3992.
- [34] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel, “mT5: A massively multilingual pre-trained text-to-text transformer,” in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, June 2021, pp. 483–498.
- [35] Zhe Zhao, Hui Chen, Jinbin Zhang, Xin Zhao, Tao Liu, Wei Lu, Xi Chen, Haotang Deng, Qi Ju, and Xiaoyong Du, “UER: An open-source toolkit for pre-training models,” 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): System Demonstrations*, Nov. 2019, pp. 241–246.

Algorithm 2 Prompt Pool Training with Rehearsal Buffer (PPT-R) for DST

Require: Frozen pretrained model f_θ , frozen encoder q_ϕ , task number T , a sequence of data $\mathcal{D} = \{\mathcal{D}_1, \dots, \mathcal{D}_T\}$, prompt pool \mathbf{P} , prompt keys \mathbf{K} , prompt number N for each task, batch size B , epochs E , a rehearsal buffer M ;

Randomly initialize \mathbf{P}, \mathbf{K} ;

for $t = 1$ to T **do**

Obtain new dataset $\mathcal{D}'_t = \mathcal{D}_t \cup M$;

Select N keys and prompts $\mathbf{K}_x = \{K_{Nt}, \dots, K_{N(t+1)-1}\}$, $\mathbf{P}_x = \{P_{Nt}, \dots, P_{N(t+1)-1}\}$;

for $e = 1$ to E **do**

Obtain a mini-batch of data $\{(x_b, y_b, \mathbf{I}(x_b \in \mathcal{D}_t))\}_{r=b}^B$ from \mathcal{D}'_t ;

Calculate the context vector $c_{x_b} = q_\phi(x_b)$ for all input samples;

Concatenate the embedding of the input sequence with the selected prompts and obtain an embedding batch $\{(E_p(x_b), y_b, \mathbf{I}(x_b \in \mathcal{D}_t))\}_{r=b}^B$;

Calculate the loss $\mathcal{L} =$

$$\frac{1}{B} \sum_{b=1}^B [CE(f_\theta(E_p(x_b)), y_b)] + \lambda \sum_{k_i \in \mathbf{K}_x} BCE(\gamma(c_{x_b}, k_i), \mathbf{I}(x_b \in \mathcal{D}_t));$$

end for

Update the selected keys \mathbf{K}_x and prompts \mathbf{P}_x using the loss \mathcal{L} ;

Update the rehearsal buffer $M = M \cup S_t$, where S_t denotes 50 dialogs randomly selected from \mathcal{D}_t ;

end for

A. TABLES AND ALGORITHMS

We present the detailed algorithm of prompt pool training with rehearsal buffer, which is shown in Algorithm 2.

The statistics of the SGD [22] dataset is shown in Table 6.

B. IMPLEMENTATION DETAILS

The pretrained encoder-decoder model f_θ is a T5-small model [21], which has 60M parameters. Besides, we choose Sentence-BERT [33] as our sentence encoder model q_ϕ , which has been found to outperform the T5 encoder in our experiments. For every task, the epoch number is set to 20, learning rate is set to 0.25 with linear decay to 0, and the weight of the key selection loss λ is set to 0.03. As for the prompt pool, the pool size $J = 150$ and the number of prompt for each task $N = 10$. Each prompt has a token length $L_p = 10$. The dimension of keys, D_K , is the same as the hidden size of Sentence-BERT, 384, and the dimension of prompts, D , is the same as the embedding size of T5-small, 512.

For experiments on Chinese dataset, we choose MT5-

Task name	train	val	test
services_4	680	97	208
flights_1	4680	667	1379
services_3	959	143	290
flights_3	420	75	116
trains_1	415	67	117
homes_2	424	56	139
rentalcars_2	631	91	185
restaurants_1	2098	297	581
music_1	468	73	142
hotels_4	559	99	141
media_2	215	29	71
hotels_3	737	100	193
rentalcars_3	332	55	99
hotels_1	868	105	250
homes_1	1829	282	540

Table 6. The number of samples in the 15 selected tasks.

small [34] as the frozen pretrained model and SBERT-Chinese [35] as the encoder model.