electronics

MDPI

Article

Prompt-Based End-to-End Cross-Domain Dialogue State Tracking

Hengtong Lu [1,*], Lucen Zhong [1], Huixing Jiang [2], Wei Chen [2], Caixia Yuan [1] and Xiaojie Wang [1,*]

[1] School of Artificial Intelligence, Beijing University of Posts and Telecommunications, Beijing 100083, China; zhonglucen@bupt.edu.cn (L.Z.); yuancx@bupt.edu.cn (C.Y.)
[2] Li Auto Inc., Beijing 101399, China; jianghuixing@lixiang.com (H.J.); chenwei10@lixiang.com (W.C.)
* Correspondence: luhengtong@bupt.edu.cn (H.L.); xjwang@bupt.edu.cn (X.W.)

Abstract: Cross-domain dialogue state tracking (DST) focuses on using labeled data from source domains to train a DST model for target domains. It is of great significance for transferring a dialogue system into new domains. Most of the existing cross-domain DST models track each slot independently, which leads to poor performances caused by not considering the correlation among different slots, as well as low efficiency of training and inference. This paper, therefore, proposes a prompt-based end-to-end cross-domain DST method for efficiently tracking all slots simultaneously. A dynamic prompt template shuffle method is proposed to alleviate the bias of the slot order, and a dynamic prompt template sampling method is proposed to alleviate the bias of the slot number, respectively. The experimental results on the MultiWOZ 2.0 and MultiWOZ 2.1 datasets show that our approach consistently outperforms the state-of-the-art baselines in all target domains and improves both training and inference efficiency by at least 5 times.

Keywords: cross-domain dialogue state tracking; prompt-based method; end-to-end method; slot correlation

Citation: Lu, H.; Zhong, L.; Jiang, H.; Chen, W.; Yuan, C.; Wang, X. Prompt-Based End-to-End Cross-Domain Dialogue State Tracking. *Electronics* **2024**, *13*, 3587. https://doi.org/10.3390/electronics13183587

Academic Editor: Domenico Ursino

Received: 20 July 2024
Revised: 30 August 2024
Accepted: 6 September 2024
Published: 10 September 2024

1. Introduction

Dialogue state tracking (DST) aims to identify user intents up to the current turn of a dialogue and compact them as a set of slot value pairs. DST is a key component in task-oriented dialogue systems. Existing DST models, such as MGCRL [1], DSETA [2], LAL-DST [3], MDST [4], DSDN [5], TS-DST [6], AG-DST [7], and TripPy [8], perform well on a given domain provided with numerous labeled training data. Since it is infeasible to create labeled data for all emerging new domains, applying the DST models trained in the known (source) domains with labeled data to a new (target) domain without labeled data, the so-called cross-domain dialogue state tracking, becomes a challenging yet attractive task.

Existing work on cross-domain dialogue state tracking is mainly classified into two categories. The first one is to build cross-domain transferable models to share knowledge between domains, such as TRADE [9], SUBMT [10], MA-DST [11], CLMQ [12], SDM [13], and CDST [14]. The second is to incorporate different slot descriptions to capture the shared information between the source domain slot and the target domain slot to facilitate cross-domain knowledge transfer SimpleTOD [15], Prompter [16], and Coref-DST [17]. The state-of-the-art models like [12,15–17] employ a pre-trained encoder–decoder backbone like GPT-2 or T5. Slot names are one part of the inputs of the models. But each time, only one slot name is used, i.e., the models track each slot independently, which suffers from two limitations. The first issue is the significant time consumption during training and inference, especially when dealing with datasets containing a large number of samples and slots. The second is that the correlation among different slots, which has been verified to be helpful for improving the performance of DST [18,19], is not taken into account.

Ideally, tracking all slots jointly and generating all values by one inference could alleviate both of the above limitations. Practically, two biases should be addressed in order

to exploit the advantages of the pre-trained encoder–decoder joint models for cross-domain dialogue state tracking. The first one is the bias of the slot order. For the joint tracking of all slots in a domain, the sequence of all slot names in a given source domain is used as a part of inputs, and the order of the slots in the sequence should be fixed through the training process. The model is, therefore, prone to learn the fixed order of slots, which is the bias of the slot order. Since the slot names in the target domain might be very different from those in source domains, the performance of a cross-domain dialogue state tracking model, therefore, suffers from the bias of the order. The second one is the bias of the slot number. When the source domains are given, the number of different slots in each domain is determined. When the slot name sequence is used as a part of inputs during training, the model is prone to learn to output the given number of slot values, which is the bias of the slot number. The bias results in the possibility of generating more or less slot values in target domains.

To address the problems above, we propose an end-to-end cross-domain joint dialogue state tracking model based on slot prompts. The model takes the slot prompt formatted by a template as a prefix of the input dialogue history and then generates corresponding slot values. During training, the dynamic shuffle method is implemented for the slot prompt template to format the slot prompt with a different slot order. As a result, the model alleviates the slot order bias and more sufficiently learns the correlation between the slots. In addition, the dynamic sampling of slot numbers is performed according to the slot prompt template, which smooths the distribution of slot numbers in the source domains and alleviates the bias of the model to slot numbers. Recently, instruction tuning methods [20–22] based on large language models have garnered attention as a novel approach to enhance zero-shot performance. Our proposed method can also be integrated with these instruction tuning approaches to improve the generalization of large language models fine-tuned on source domain data in target domains. To validate this, we conducted experiments using several large language models of varying performances and sizes.

Our main contributions are as follows:

- We propose a Prompt-based end-to-end Cross-domain joint dialogue state tracking (PCDST) model. To the best of our knowledge, it is the first study on an end-to-end multi-slot joint modeling method for cross-domain dialogue state tracking.
- We propose a dynamic shuffle prompt template construction method, which enriches the diversity of prompt templates, and a dynamic sampling template construction method, which smooths the distribution of slot numbers in the source domain to alleviate the data bias to the slot order and slot number, respectively.
- The experimental results on MultiWOZ 2.0 and MultiWOZ 2.1 based on T5 show that our model consistently outperforms the SOTA baseline model in all target domains, and it improves the efficiency of training and inference by at least 5 times.
- Furthermore, we integrated our approach with instruction fine-tuning methods based on large language models, showing that our method enhances performance in target domains across various models of different performances and sizes.

The remainder of this paper is organized as follows: In Section 2, we review the related work on cross-domain dialogue state tracking, joint DST of multiple slots, and prompt-based learning in dialogue. Section 3 introduces our proposed method, including task formulation, the PCDST approach, and the process of prompt formation for both training and inference. Section 4 presents our experimental setup, including datasets, baselines, and implementation details. In Section 5, we discuss the results, including main findings, training and inference time, performance on large language models, and ablation studies. Finally, Section 6 concludes the paper. For reproducibility, our code for this paper will be published at https://github.com/luhengtong/CDST.git.

2. Related Work

2.1. Cross-Domain Dialogue State Tracking

There are mainly two streams of methods in previous cross-domain dialogue state tracking studies. The first is to build cross-domain transferable models to share knowledge between domains. Wu et al. [9] first utilized the copy mechanism to generate dialogue states from dialogue history and reduce the model's dependence on ontology knowledge. Lee et al. [10] matched the history attendance on the slot name with the slot values and predicted the slot value with a non-parametric method. Gao et al. [23] formulated DST as a machine reading problem and converted the slot name into a natural-language problem to transfer between different domains. Li et al. [12] proposed an ontology-free conditional language modeling framework for dialog state tracking via generative question answering. The second is to incorporate a different slot description that captures the shared information between the source domain slot and the target domain slot to facilitate cross-domain knowledge transfer. Lin et al. [15] investigated the effectiveness of different slot description formulations and proposed slot-type informed descriptions that capture the shared information of different slots.

All these models track different slots independently. In contrast, our model tracks all slots in a domain jointly. It can not only learn the correlation between different slots but also reduce the time complexity.

Recently, with the rise of research on large models, some works [24,25] have utilized large models for cross-domain dialogue state tracking. These methods leverage powerful, yet proprietary, language models like Codex-Davinci-002 and ChatGPT for cross-domain dialogue state tracking, which makes training and inference based on these models difficult and costly due to their closed-source nature. In our work, we integrated our methods with instruction fine-tuning on open-source large language models, achieving around 90% performance.

2.2. Joint DST of Multiple Slots

Joint DST for multiple slots is widely used in single and multi-domain dialogue tasks, which is divided into two categories. One is to use the graph structure to explicitly model the correlation between slots in different domains [18]. Another category is sequence-to-sequence dialogue state generation [19,26]. Hosseini-Asl et al. [19] proposed a unified approach to task-oriented dialogue that employs a single causal language model to perform sequence prediction in DST, Policy, and NLG (nature language generation) based on history. Yang et al. [26] fine-tuned GPT-2 on the sequence of the entire dialog session consisting of user utterance, belief state, database result, system act, and system response of every dialog turn. Seq2Seq-DU [27] encoded schema information and history flatly and generated pointers instead of tokens in the decoder.

All the above methods are for multi-domain DST, which cannot be directly applied to cross-domain dialogue state tracking because of different settings between multi-domain and cross-domain dialogue state tracking. Multi-domain DST models can only generate slots seen in the training domain, while slot names in target domains in a cross-domain dialogue state tracking setting might be unseen in the source domain training data.

2.3. Prompt-Based Learning in Dialogue

Prompts are a recent research methodology for extending language models' knowledge. Prompting-based learning is also used to improve the model performance in dialogue-related tasks. Lee et al. [28] designed a schema-driven prompting to provide task-aware history encoding for each slot in multi-domain DST. Su et al. [29] introduced a new dialogue multi-task pre-training strategy that designs a task-specific prompt for NLU (nature language understanding), DST, Policy, and NLG, respectively, allowing for the model to learn the primary TOD (task-oriented dialogue) task completion skills from heterogeneous dialogue corpora. Mi et al. [30] added additional task-specific definitions and constraints to the prompt to enrich the prompt information on NLU, DST, and NLG tasks.

3. Method

3.1. Task Formulation

Given n source domains $D_{src} = \{D_1, D_2, \cdots, D_n\}$ and one target domain D_{tgt}, each domain $D_i (i \in \{1, 2, \cdots, n, tgt\})$ has a corresponding slot set $S_i = \{s_1, s_2, \cdots, s_{n_i}\}$, where n_i is the slot number of the domain D_i. For a sample from domain D_i, let C_i^j represent the dialogue history, which is a sequence of alternating user utterances and system responses: $C_i^j = \{u_1, a_1, u_2, a_2, \cdots, u_l, a_l, \cdots, u_{t-1}, a_{t-1}, u_t\}$. Here, u_t denotes the user's tth utterance, and a_t denotes the corresponding system response. The corresponding dialogue state is denoted as $B_i^j = \{(s_1, v_1), (s_2, v_2), \cdots, (s_m, v_m), \cdots, (s_{n_i}, v_{n_i})\}$, where $s_m \in S_i$ represents the mth slot in domain D_i, and $v_m \in V_m$ is the value associated with slot s_m. V_m is the set of possible values for slot s_m.

Cross-domain DST uses the data pairs (C_i^j, B_i^j) in the source domains to train a DST model and uses it to track the dialogue state B_{tgt}^j of dialogue history C_{tgt}^j from target domain D_{tgt}.

3.2. PCDST

As shown is Figure 1, our proposed model PCDST takes the slot prompt PS and dialogue history C as inputs, and it outputs the corresponding slot value prompt PV. Specifically, we use special tokens that represent the speaker information to concatenate the utterances, i.e., $[speaker1][S][speaker2]u_1[speaker1]a_2 \cdots [speaker2]u_n$, where $[speaker1]$ represents the system, and $[speaker2]$ represents the user. For training, we format the corresponding slot prompt sequence PS_i^j and slot value prompt sequence PV_i^j according to B_i^j (the specific formation method is given in Section 3.3). PS_i^j and dialogue history C_i^j are concatenated into a single sequence, which is used as the input to the encoder, and the decoder generates the corresponding slot value prompt sequence PV_i^j. The learning object of our model is minimizing the negative log-likelihood of PV_i^j given C_i^j and PS_i^j, that is,

$$\mathcal{L} = -\sum_{(i,j)} \log p\left(PV_i^j \mid PS_i^j, C_i^j\right) \tag{1}$$

When testing on the target domain, we use the slot set S_{tgt} of the target domain to format the slot prompt sequence PS_{tgt} (the specific formation method is given in Section 3.4); the model takes as input a slot prompt in the target domain PS_{tgt} and C_{tgt}^j and outputs slot value prompt PS_{tgt}^j. Then, we can combine PS_{tgt} and PV_{tgt}^j to obtain the target dialogue state B_{tgt}^j.

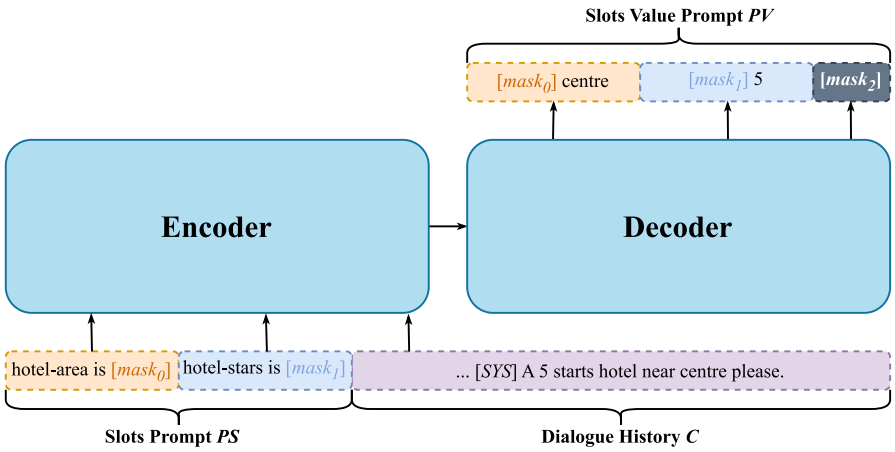


Figure 1. Illustration of our proposed model PCDST.

3.3. Prompt Formation for Training

As shown in Figure 2, we first construct the prompt template, and then we format the slot prompt and slot value prompt according to the prompt template. Different construction methods are applied to enrich the diversity of the templates. The prompt formation method and prompt template construction methods proposed by us are as follows.

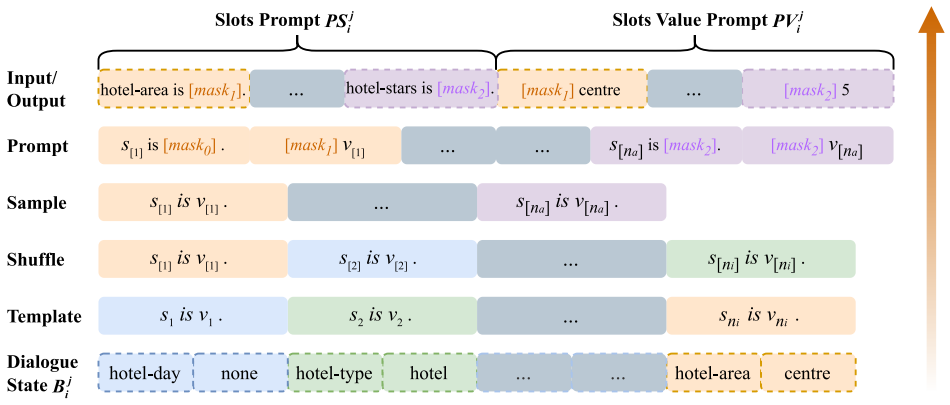


Figure 2. Prompt formation for training.

3.3.1. Prompt Template Construction

To begin with, each slot name and its corresponding value are joined into a sequence. Subsequently, all generated sequences from different slot value pairs are concatenated, guided by a specific slot order. For semantic coherence, some delimiters are added during sequence concatenation. In detail, "is" joins slot names and values, meanwhile, "." concatenates slot value pairs. The sequence after construction is used as the template T_i to format the slot prompt and slot value prompt.

$$T_i^j = s_1 \text{ is } v_1 \text{ . } \cdots \text{ . } s_m \text{ is } v_m \text{ . } \cdots \text{ . } s_{n_i} \text{ is } v_{n_i} \text{ .} \tag{2}$$

where we denote s_m and v_m as a slot value pair sub-utterance.

3.3.2. Shuffle Prompt Template Construction

In order to prevent the model from learning the bias about the slot order of the source domain, we propose a dynamic shuffle method for templates to shuffle the position of the slot pair sub-utterances.

$$\begin{aligned} &\text{Shuffle}\left(T_i^j\right) \\ &= \text{Shuffle}(s_1 \text{ is } v_1 \text{ . } \cdots s_m \text{ is } v_m \text{ . } \cdots s_{n_i} \text{ is } v_{n_i} \text{ .}) \\ &= s_{\sigma(1)} \text{ is } v_{\sigma(1)} \text{ . } \cdots s_{\sigma(m)} \text{ is } v_{\sigma(m)} \text{ . } \cdots s_{\sigma(n_i)} \text{ is } v_{\sigma(n_i)} \text{ .} \end{aligned} \tag{3}$$

where $s_{\sigma(m)}$ represents the m-th slot in the dialogue state sequence, and σ is the permutation function.

3.3.3. Sample Prompt Template Construction

In order to smooth the distribution of the slot number in the slot prompt in the source domain, we propose a dynamic sample method for templates. Specifically, we sample a slot number n_a according to a distribution $p(n_a)$, and then we truncate the template by the number n_a to hold the first n_a slot value pair sub-utterances. The distribution of the slot number in the examples of domain D_i before the sample is as follows:

$$p(n_a) = \begin{cases} 1 & n_a = n_i \\ 0 & n_a \neq n_i \end{cases} \tag{4}$$

The sampling distribution of the number n_a is

$$p(n_a) = \begin{cases} 1-\alpha, & n_a = n_i \\ \frac{\alpha}{n_i - 1}, & n_a < n_i \\ 0, & n_a > n_i \end{cases} \tag{5}$$

where α is a smoothing factor for the distribution of the slot number in the source domain, $0 \leq \alpha \leq 1$, which represents the sum of the proportions of the examples that contain less than n_i slots during training.

$$\begin{aligned} &\text{Sample}\left(T_i^j\right) \\ &= \text{Sample}(s_1 \text{ is } v_1 \text{ . } \cdots s_m \text{ is } v_m \text{ . } \cdots s_{n_i} \text{ is } v_{n_i} \text{ .}) \\ &= s_1 \text{ is } v_1 \text{ . } \cdots s_{n_a} \text{ is } v_{n_a} \text{ .} \end{aligned} \tag{6}$$

3.3.4. Slot Prompt and Slot Value Prompt Formation

To format the slot prompt and slot value prompt according to the template T_i^j, we mask the consecutive spans corresponding to the slot value words in the template with different specific mask tokens to format the slot prompt PS_i^j , and we concatenate the masked slot value words with mask tokens to format the slot value prompt PV_i^j.

$$PS_i^j = s_1 \text{ is } [\text{mask}_1] \text{ . } s_2 \text{ is } [\text{mask}_2] \text{ . } \cdots s_m \text{ is } [\text{mask}_\text{m}] \text{ . } \cdots s_{n_i} \text{ is } [\text{mask}_{\text{n}_i}] \text{ .}$$

$$PV_i^j = [\text{mask}_1] v_1 [\text{mask}_2] v_2 \cdots [\text{mask}_\text{m}] v_m s_m \cdots [\text{mask}_{\text{n}_i}] v_{n_i} \left[\text{mask}_{\text{n}_{\text{i}+1}}\right]$$

where $[mask_m]$ is a special token representing the slot value mask, which is used as the m-th mask token that appears in the template sequence, and $[mask_{n_{i+1}}]$ is added to the end of the slot value prompt sequence to mark the end of decoding. Different source domains and target domains share the same mask tokens set. As shown in Figure 2, our model first generates a prompt template T_i^j according to the dialogue state $B_i^j = \{(s_1, v_1), \cdots, (s_m, v_m), \cdots, (s_{n_i}, v_{n_i})\}$ when formatting prompts for training, and then it shuffles and samples the template T_i^j, and finally, it formats slot prompts and slot value prompts based on the obtained prompt template.

$$PS_i^j = s_{\sigma(1)} \text{ is } [\text{mask}_1] \text{ . } s_{\sigma(2)} \text{ is } [\text{mask}_2] \text{ . } \cdots s_{\sigma(m)} \text{ is } [\text{mask}_\text{m}] \text{ . } \cdots s_{\sigma(n_a)} \text{ is } [\text{mask}_{\text{n}_\text{a}}] \text{ .}$$

$$PV_i^j = [\text{mask}_1] v_{\sigma(1)} [\text{mask}_2] v_{\sigma(2)} \cdots [\text{mask}_\text{m}] v_{\sigma(m)} \cdots \left[\text{mask}_{[\text{n}_\text{a}]}\right] v_{\sigma(n_a)} \left[\text{mask}_{\text{n}_{\text{a}+1}}\right]$$

where σ is the permutation function.

3.4. Prompt Formation for Inference

During testing, given a slot set in the target domain $S_{tgt} = s_1, s_2, \cdots, s_{n_{tgt}}$, we sort the slots in the set S_{tgt} in a specific order to obtain a list of slots, and we use the connective words "is", "." and mask token to concatenate the slots in the slot list to obtain the slot prompt sequence.

$$PS_i^j = s_1 \text{ is } [\text{mask}_1] \text{ . } s_2 \text{ is } [\text{mask}_2] \text{ . } \cdots s_m \text{ is } [\text{mask}_\text{m}] \text{ . } \cdots s_{n_{tgt}} \text{ is } \left[\text{mask}_{\text{n}_{tgt}}\right] \text{ .}$$

Likewise, we input the dialogue history C_{tgt}^j from the target domain and PS_{tgt} to generate the slot value prompt sequence

$$PV_i^j = [\text{mask}_1] v_1 [\text{mask}_2] v_2 \cdots [\text{mask}_\text{m}] v_m s_m \cdots \left[\text{mask}_{\text{n}_{\text{tgt}}}\right] v_{n_{tgt}} \left[\text{mask}_{\text{n}_{\text{tgt}+1}}\right]$$

Then, we obtain the dialogue state B^j_{tgt} in the target domain according to PS_{tgt} and PV^j_{tgt}.

$$B^j_{tgt} = (s_1, v_1), (s_2, v_2), \cdots, (s_m, v_m), \cdots, (s_{n_{tgt}}, v_{n_{tgt}})$$

4. Experiments

4.1. Datasets and Evaluation

MultiWOZ is one of the most widely used benchmarks for dialogue state tracking. MultiWOZ 2.0 [31] contains over 10K task-oriented dialogues. MultiWOZ 2.1 [32] improves version 2.0 by fixing 32% of dialogue state annotations across 40% of the turns. We use both of them to compare with the existing work. We standardize the labeled information following the work of [15] and remove the dialogues containing only two domains (police and hospital) from the training set. The final data contain five domains (attraction, hotel, restaurant, train, taxi), covering 30 domain–slot pairs.

In the cross-domain zero-shot DST setting, the model randomly takes one domain as the target domain and the remaining four domains as the source domains. After training on the labeled data from the source domains, the model is used and tested in the target domain. The Joint Goal Accuracy (JGA) is used to evaluate the DST performance, which measures the proportion of turns with correctly predicted dialogue states. The generated dialogue states are considered to be correct if and only if all of the predicted values exactly match the oracle values.

4.2. Baselines

TRADE [9] A transferable dialogue state generator that utilizes a copy mechanism to facilitate domain knowledge transfer.

MA-DST [11] A DST model that improves TRADE by utilizing multi-layer cross attention to fuse the history and slot name.

SUMBT [10] A slot-utterance matching DST model based on the BERT.

SUMBT-variant [33] A variant of SUMBT introducing attention modulation to improve the cross-domain zero-shot performance of SUMBT.

SimpleTOD++ [19] A variant of SimpleTOD [15] that uses GPT2 [34] to generate the dialogue states.

T5DST [15] A dialogue state generator based on T5 [35], which encodes the slot description and history to generate corresponding slot values.

TransferQA [36] A cross-task zero-shot DST method where the model is initially pre-trained on question answering data and then applied to unseen domains.

SDM [13] A DST model that leverages slot prompts, slot values demonstration, and slot constraint object to effectively capture slot–slot, slot–value, and slot–context dependencies, improving zero-shot/few-shot DST performance.

CLMQ [12] An ontology-free conditional language modeling framework for dialog state tracking via generative question answering.

Prompter [16] A DST model that uses target domain slot descriptions to generate dynamic prefixes for self-attention mechanisms, enabling effective zero-shot DST.

4.3. Implementation Details

We implemented PCDST based on T5-small and T5-base [35], respectively. The masked tokens in T5 $\{< extra\_id\_0 >, < extra\_id\_1 >, \cdots, < extra\_id\_11 >\}$ were reused as mask tokens $[mask_m]$ in our model. Intuitively, special tokens $[speaker1]$ and $[speaker2]$ were added to capture speakers' information. We trained all models for 3 epochs with a batch size of 8 on one Nvidia V100 GPU, using AdamW [37] with a base learning rate of 1e-4 for T5-small and 5e-5 for T5-base. The smoothing factor α was set to 0.2 and 0.6 for T5-small and T5-base, respectively. We evaluated every 1000 steps and chose the model with the best JGA performance on its validation set as the final model to be tested on the target domain. All predictions were made using greedy decoding.

5. Results and Discussions

5.1. Main Results

The proposed PCDST was evaluated on both MultiWOZ 2.0 and MultiWOZ 2.1. Since SOTA models on these two datasets are different models with different sizes of parameters, to make a parallel comparison, we utilized T5-small and T5-base for MultiWOZ 2.0 and MultiWOZ 2.1, respectively. The experimental results on these two datasets are shown in Table 1 and Table 2, respectively.

Table 1. Comparison of results on MultiWOZ 2.0.

Model	Base Model	Avg	Att	Hot	Res	Tax	Tra
TRADE †	N	25.61	19.87	13.7	11.52	60.58	22.37
SUMBT ‡	N	27.40	23.57	14.51	17.19	60.41	21.31
SUMBT-variant ‡	N	29.24	29.83	17.09	16.8	59.72	22.74
SimpleTOD++ †	GPT2-b	29.65 ± 0.58	28.01 ± 1.30	17.69 ± 1.00	15.57 ± 1.54	59.22 ± 0.95	27.75 ± 1.16
T5DST †	T5-s	35.2 ± 0.59	33.09 ± 1.60	21.21 ± 0.61	21.65 ± 1.07	64.62 ± 0.55	35.42 ± 1.42
PCDST	T5-s	**36.88 ± 0.86**	**37.31 ± 0.51**	**24.77 ± 0.5**	**23.79 ± 1.34**	64.23 ± 0.62	34.29 ± 1.84

We ran each experiment three times with different random seeds, and we report the mean and standard deviation here. † Result from [15]. ‡ Result from [33]. -s means -small. -b means -base.

Table 2. Comparison of results on MultiWOZ 2.1.

Model	Base Model	Avg	Att	Hot	Res	Tax	Tra
TRADE †	N	25.69	20.06	14.2	12.59	59.21	22.39
MA-DST †	N	26.87	22.46	16.28	13.56	59.27	22.76
SUMBT †	Bert-b	28.18	22.6	19.8	16.5	59.5	22.5
T5DST ‡	T5-s	33.56	31.92	20.72	20.09	64.12	28.83
T5DST ‡	T5-b	36.25	35.51	22.48	25.04	65.93	34.82
TransferQA ‡	T5-l	35.77	31.25	22.72	26.28	61.87	36.72
SDM ‡	T5-s	35.55	33.92	19.85	20.75	66.25	36.96
SDM ‡	T5-b	40.18	37.83	26.50	27.05	69.23	40.27
CLMQ †	GPT2-b	36.02	34.3	22.94	24.65	59.68	38.55
CLMQ †	GPT2-m	39.27	42.39	24.88	27.69	60.32	41.05
Prompter *	PPTOD-s	**37.27 ± 7.0**	**35.80 ± 0.7**	**19.20 ± 0.8**	**26.00 ± 0.7**	**66.30 ± 0.2**	**39.00 ± 0.5**
PCDST	T5-b	**42.62 ± 0.62**	**46.45 ± 0.38**	**25.94 ± 0.26**	**30.85 ± 0.63**	**67.98 ± 0.58**	**41.91 ± 2.12**

We ran each experiment three times with different random seeds, and we report the mean and standard deviation here. † Result from [12]. ‡ Result from [13]. * Result from [16]. -s means -small. -b means -base. -m means -medium. -l means -large.

The proposed PCDST outperformed all compared methods almost in all target domains in terms of joint goal accuracy by a large margin. Specifically, on MultiWOZ 2.0, the proposed PCDST with T5-small achieved 1.6% improvement on average compared with T5DST (T5-small), and on MultiWOZ 2.1, the proposed PCDST with T5-base achieved an overall 3.35% improvement compared with CLMQ (GPT2-medium). The above results indicate that our proposed model can effectively learn knowledge from the source domains that is beneficial for the target domain.

5.2. Training and Inference Time

To intensively investigate the computational complexity of different models, the comparative experiments of training and inference time were conducted, respectively. The elapsed times for each model with training 1 epoch are shown in Table 3. As shown in Table 3, the elapsed time of training PCDST with T5-small was merely 18.69% of that by T5DST (T5-small). Meanwhile, the elapsed time of training PCDST with T5-base was 18.32% of that by CLMQ (GPT2-base). The average inference times of one sample for each domain are shown in Table 3. The results are consistent with those of training time. The average

inference time taken by PCDST with T5-small was only 13.36% of that by T5DST (T5-small). Meanwhile, PCDST with T5-base took only 18.0% of the time taken by CLMQ (GPT2-base). The experimental results show that the proposed PCDST is at least five times more efficient than the previous SOTA models.

Table 3. The elapsed time (minutes) for each model with training for 1 epoch and average inference time (milliseconds) of 1; example on the (Att)raction, (Hot)el, (Res)taurant, (Tax)i, and (Tra)in domains.

	Training Time (Minutes)						**Inference Time (Milliseconds)**					
Model	**Avg**	**Att**	**Hot**	**Res**	**Tax**	**Tra**	**Avg**	**Att**	**Hot**	**Res**	**Tax**	**Tra**
CLMQ-b	326	335	265	310	380	340	78	43	110	97	45	95
T5DST-s	139	165	115	135	150	130	375	179	583	440	244	428
T5DST-b	409	450	305	365	510	415	125	67	187	131	88	153
PCDST-s	26	25	25	25	30	25	10	6	20	9	8	10
PCDST-b	69	70	65	65	75	70	23	17	28	22	22	24

5.3. Results on Large Language Models Instruction Tuning

To validate the effectiveness of our proposed method combined with instruction tuning, we conducted experiments using several open-source large language models of different sizes and capabilities. We selected models including Llama1-7b [38], Llama2-7b, Llama2-13b [39], Llama3-8b [40], Gemma-2b [41], and Gemma2-9b [42].

Initially, we fine-tuned these large language models based on source domain data, utilizing the following instruction construction format for instruction tuning (which is denoted as IT):

Please follow the format of "<PAN> slot name <PAV> slot value" to extract the slot value existing in the dialogue.
The slot name to be extracted is "{slot_names}".
The dialog history is
"'{dial_history}"'
Among them, "<U>" means User, "<A>" means Agent.
If a slot does not exist in the dialog, the slot value is replaced by None.
Please follow the format of "<PAN> slot name "<PAV> slot value".
The extraction result is as follows:

Here, *{slot_names}* is a placeholder representing domain slot names, and *{dial_history}* is a placeholder representing dialogue history.

Subsequently, we integrated our method with instruction tuning (which is denoted as PCDST-IT), dynamically adjusting the number and order of slots during the tuning process. To verify the effectiveness of our approach, we tested the fine-tuned models with two types of instructions: Instruction Test_A, where *{slot_names}* in all sample instructions were sorted alphabetically, and Instruction Text_B, where the order of *{slot_names}* was randomized. The specific instruction tuning training example and text instruction is shown in Appendix A. Specific parameters used for model fine-tuning and the computational resources employed are detailed in Appendix B. The experimental results are presented in Table 4. Detailed experimental results are presented in Appendix C.

As shown in Table 4, combining our method with instruction fine-tuning consistently improved performance across all models. Comparisons between the performance of Test_A and Test_B indicate that instruction-based tuning alleviates the biases related to slot order and number. By integrating our method with instruction tuning, we mitigated these biases, thereby reducing the performance disparity between Instructions Test_A and Test_B.

Table 4. Results based on large language model instruction tuning on MultiWOZ 2.1.

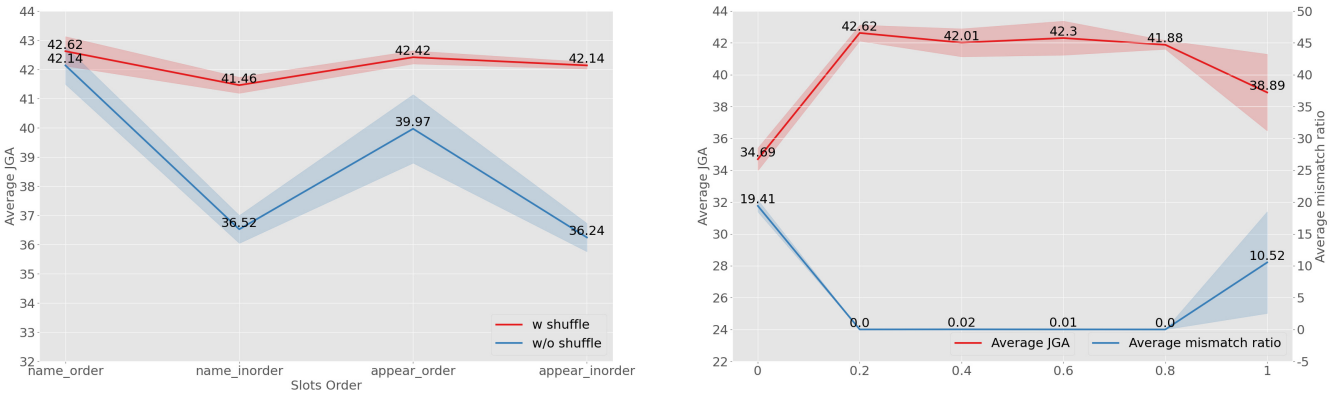
Model	Test	Llama1-7b	Llama2-7b	Llama2-13b	Llama3-8b	Gemma-2b	Gemma2-9b
IT	Test_A	44.54 ± 0.67	48.97 ± 0.16	49.21 ± 1.4	49.74 ± 0.34	41.9 ± 1.99	52.47 ± 0.59
	Test_B	40.45 ± 1.39	47.79 ± 0.32	48.61 ± 0.19	47.37 ± 1.14	40.72 ± 0.93	51.26 ± 1.63
PCDST-IT	Test_A	47.14 ± 1.73	50.76 ± 1.15	50.26 ± 0.26	50.98 ± 0.68	43.15 ± 2.48	53.08 ± 2.74
	Test_B	46.39 ± 1.2	50.87 ± 0.93	50.06 ± 0.87	50.73 ± 0.51	43.03 ± 2.1	52.46 ± 2.67

We ran each experiment three times with different random seeds, and we reported the mean and standard deviation.

5.4. Ablation Studies

5.4.1. Effect of Shuffle Construction

To explore the impact of shuffle construction for prompt templates, an ablative experiment for shuffle construction was conducted. As shown in Table 5, PCDST without shuffle construction drops 0.48% JGA on average. In addition, to discuss the effect of different slot order biasses of shuffle construction, PCSDST and PCSDST without shuffle construction were tested with four different slot orders during the inference, respectively, as shown in Figure 3a. Figure 3a shows that the performance of PCDST is more stable than PCDST without shuffle construction. Thus, the shuffle construction can effectively alleviate the learning bias of a static slot order as input from the source domains. As shown in Table 6, we show the *appear_order* in each domain.



(a) Effect of shuffle construction

(b) Effect of sampling construction

Figure 3. (**a**) The average JGA on 4 slot prompts with different slot orders. *name_order* and *name_inorder* denote that slots are arranged in alphabetic order and reverse order, respectively. *appear_order* and *appear_inorder* denote that slots are arranged in arisen order and reverse order in the dataset, respectively. (**b**) The average JGA and average mismatch ratio of the model with a different α in the sample.

Table 5. The performance of the model when the sample or shuffle is removed or both are removed.

Model	Average	Attraction	Hotel	Restaurant	Taxi	Train
PCDST	42.62 ± 0.62	46.45 ± 0.38	25.94 ± 0.26	30.85 ± 0.63	67.98 ± 0.58	41.91 ± 2.12
w/o sample	34.69 ± 0.88	45.1 ± 1.58	26.88 ± 0.31	30.24 ± 1.78	68.66 ± 0.17	2.58 ± 4.37
w/o shuffle	42.14 ± 0.79	45.02 ± 1.79	25.12 ± 0.51	29.73 ± 2.65	67.3 ± 0.32	43.52 ± 0.48
w/o all	34.16 ± 0.11	46.61 ± 2.21	25.22 ± 0.3	31.3 ± 0.8	67.05 ± 1.36	0.6 ± 1.05

Table 6. The slot order of *appear_order* in each domain, which is counted on the validation set of each domain.

Domain	Slot Order
Attraction	area, type, name
Hotel	type, name, area, pricerange, stars, internet, parking, day, people, stay
Restaurant	area, food, pricerange, name, day, people, time
Taxi	departure, destination, leaveat, arriveby
Train	destination, departure, day, arriveby, leaveat, people

5.4.2. Effect of Sampling Construction

Considering the effect of sampling construction for prompt templates, an ablative experiment for sampling construction was also conducted. As shown in Table 5, PCDST without sampling construction dropped 7.93% on average in terms of JGA, which demonstrates the effectiveness of the sampling construction.

To understand the impact of factor α in sample strategy, experiments with factor α ranging from 0 to 1 were conducted. The result of the average JGA and average mismatch ratio are shown in Figure 3b, where the mismatch ratio measures the proportion of turns with incorrect slots quantitatively. Figure 3b shows that when factor α varies from 0.2 to 0.8, the model achieves comparable performance on average JGA and generates the correct number of slots. Thus, the sampling construction can reduce the learning bias to a particular number of slots from source domains.

5.5. Case Study

Considering the importance of the correlation between slots for cross-domain dialogue state tracking, we display an example of two similar dialogue snippets from the source domain and target domain as shown in Figure 4. The example shows that the correlation between the "type" slot and the "area" slot in the hotel (source) domain is similar to the correlation between the "type" slot and the "area" slot in the attraction (target) domain. Such slot co-occurrence is a significant clue to inference target domain DST. The proposed PCDST can utilize this co-occurrence to generate correct slot values in the target domain compared with the baseline model (T5DST).

Model	Domain	Input and Output Sequence
PCDST	Source (Hotel)	hotel-area is <extra_id_0> . hotel-day is <extra_id_1> . hotel-type is <extra_id_2> . [speaker2] I'd like a hotel in the east part of the city .
		Label: <extra_id_0> east <extra_id_1> none <extra_id_2> hotel <extra_id_3>
PCDST	Target (Attraction)	attraction-area is <extra_id_0> attraction-name is <extra_id_1>. attraction-type is <extra_id_2> . [speaker2] I'm looking for a theatre to visit in the centre of Cambridge . Can you help me with this ?
		Prediction: <extra_id_0> centre <extra_id_1> none <extra_id_2> theatre <extra_id_3>
T5DST	Target (Attraction)	User: I'm looking for a theatre to visit in the centre of Cambridge . Can you help me with this ? [SEP] hotel-type
		Prediction: None

Figure 4. The examples generated by PCDST and T5DST. For ease of reading, we mark the slots in blue and the values in red.

6. Conclusions and Future Work

In this paper, we propose a prompt-based end-to-end cross-domain dialogue state tracking model for efficiently tracking all slots simultaneously. In addition, to alleviate the bias of the slot order and the slot number, we utilized a dynamic prompt template shuffle method and a dynamic prompt template sampling method, respectively. The exper-

imental results on MultiWOZ 2.0 and MultiWOZ 2.1 showed that our model consistently outperforms the SOTA baseline model in all target domains and improves the efficiency of training and inference by at least five times.

However, there are still some issues unresolved: (1) more datasets: we need to evaluate our model on more datasets to test the generalization of our framework. (2) Effect of domain similarity: we want to explore the effect of domain similarity on the scalability of our model, which requires data from more diverse domains. In future work, we will conduct research on data from domains with lower similarity and propose a more robust cross-domain DST model.

Author Contributions: H.L.: conceptualization, methodology, coding, validation, investigation, data curation, writing—original draft preparation; L.Z.: coding, validation, investigation, data curation; H.J. and W.C.: conceptualization, methodology, writing—review and editing; C.Y. and X.W.: methodology, writing—review and editing. All authors have read and agreed to the published version of the manuscript.

Funding: The work was partially supported by the National Natural Science Foundation of China (NSFC62076032).

Data Availability Statement: The datasets utilized in this study are based on publicly available datasets, and interested researchers can access them through the provided references.

Acknowledgments: The authors would like to thank Chenxu Lv and Keqing He for their contribution to label collection and valuable comments.

Conflicts of Interest: Authors Huixing Jiang and Wei Chen were employed by the company Li Auto Inc. The remaining authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

Abbreviations

The following abbreviations are used in this manuscript:

DST	Dialogue state
TOD	Task-oriented dialogue
NLU	Nature language understanding
NLG	Nature language generation

Appendix A. Instruction Tuning Example

Table A1 shows the construction of instruction input and output for a sample in the attraction domain during training and Test_B, where the construction of the input and output for Test_A is consistent with that during training.

Table A1. Instruction tuning example.

Slot Names		**[Attraction-Area, Attraction-Name, Attraction-Type]**
Dialogue history		User: Looking for places to go in town that are theatres . System: We have five theatres to choose from , is there a preferred area ? User: No , I have no preference , which one do you recommend ?
Trainging instruction	Input	Please follow the format of “<PAN> slot name <PAV> slot value” to extract the slot value existing in the dialogue. The slot name to be extracted is “<PAN> attraction-area <PAN> attraction-name <PAN> attraction-type”. The dialog history is ‘‘‘ <U> Looking for places to go in town that are theatres . <A> We have five theatres to choose from , is there a preferred area ? <U> No , I have no preference , which one do you recommend ? ’’’ Among them, “<U>” means User, “<A>” means Agent. If a slot does not exist in the dialog, the slot value is replaced by None. Please follow the format of “<PAN> slot name “<PAV> slot value". The extraction result is as follows:
	Output	<PAN> attraction-area <PAV> dontcare <PAN> attraction-name <PAV> none <PAN> attraction-type <PAV> theatre
Instruction Text_B	Input	Please follow the format of “<PAN> slot name <PAV> slot value” to extract the slot value existing in the dialogue. The slot name to be extracted is “<PAN> attraction-name <PAN> attraction-type <PAN> attraction-area”. The dialog history is ‘‘‘ <U> Looking for places to go in town that are theatres . <A> We have five theatres to choose from , is there a preferred area ? <U> No , I have no preference , which one do you recommend ? ’’’ Among them, “<U>” means User, “<A>” means Agent. If a slot does not exist in the dialog, the slot value is replaced by None. Please follow the format of “<PAN> slot name “<PAV> slot value”. The extraction result is as follows:
	Output	<PAN> attraction-name <PAV> none <PAN> attraction-type <PAV> theatre <PAN> attraction-area <PAV> dontcare

Appendix B. Implementation Details

We trained all models on eight Nvidia A800 GPUs. The detailed training parameters for every model are displayed in Table A2, where the batch size is represented as “per GPU training batch size X GPU numbers × gradient accumulation steps”.

Table A2. Detailed training parameters for every model.

	Llama1-7b	**Llama2-7b**	**Llama2-13b**	**Llama3-8b**	**Gemma-2b**	**Gemma2-9b**
learning rate	5e-6	5e-6	5e-6	5e-6	5e-6	5e-6
batch size	20 × 8 × 1	20 × 8 × 1	16 × 8 × 1	20 × 8 × 1	16 × 8 × 2	8 × 8 × 4
epochs	3	3	3	3	2	2
smoothing factor	0.6	0.6	0.6	0.6	0.6	0.6

Appendix C. Detailed Results

Detailed results of each domain based on large language model instruction tuning on MultiWOZ 2.1 are shown in Table A3.

Table A3. Detailed results based on large language model instruction tuning on MultiWOZ 2.1.

Base_Model		Average	Attraction	Hotel	Restaurant	Taxi	Train
Llama1-7b	IT	44.54 ± 0.67	46.63 ± 7.43	28.14 ± 1.81	48.65 ± 0.59	66.2 ± 2.13	33.06 ± 3.31
		40.45 ± 1.39	42.14 ± 11.5	27.54 ± 0.95	39.24 ± 2.43	63.59 ± 1.88	29.71 ± 1.61
	PCDST-IT	47.14 ± 1.73	52.36 ± 2.82	30.42 ± 1.67	48.62 ± 4.22	69.36 ± 0.23	34.96 ± 3.55
		46.39 ± 1.2	52.8 ± 3.16	30.09 ± 1.02	45.57 ± 2.81	68.53 ± 0.28	34.98 ± 2.33
Llama2-7b	IT	48.97 ± 0.16	54.46 ± 1.95	32.41 ± 0.53	59 ± 0.28	69.04 ± 1.72	29.92 ± 1.2
		47.79 ± 0.32	54.28 ± 1.18	31.24 ± 0.43	57.25 ± 1.36	65.73 ± 0.73	30.46 ± 0.89
	PCDST-IT	50.76 ± 1.15	56.9 ± 1.38	31.87 ± 4	59.52 ± 3.3	69.7 ± 1.18	35.8 ± 1.28
		50.87 ± 0.93	56.4 ± 1.74	32.69 ± 2.13	59.34 ± 3.26	69.04 ± 1	36.87 ± 2.85
Llama2-13b	IT	49.21 ± 1.4	59.08 ± 0.85	34.46 ± 0.01	54.63 ± 1.06	69.85 ± 9.9	28.02 ± 45.86
		48.61 ± 0.19	57.16 ± 2.05	32.77 ± 0.24	53.36 ± 1.82	68.24 ± 2.78	31.54 ± 2.78
	PCDST-IT	50.26 ± 0.26	59.27 ± 0.51	32.69 ± 0.56	50.82 ± 1.85	70.1 ± 0.39	38.44 ± 1.7
		50.06 ± 0.87	59.41 ± 1.26	31.52 ± 1.89	49.52 ± 1.32	70.36 ± 0.56	39.49 ± 13.11
Llama3-8b	IT	49.74 ± 0.34	60.82 ± 1.16	32.57 ± 2.59	54.5 ± 1.84	69.21 ± 0.92	31.61 ± 1.24
		47.37 ± 1.14	57.01 ± 1.11	26.63 ± 9.03	55.74 ± 2.25	66.37 ± 0.99	31.08 ± 1.53
	PCDST-IT	50.98 ± 0.68	59.14 ± 2.57	34.58 ± 1.89	53.19 ± 2.31	71.31 ± 2.17	36.7 ± 1.39
		50.73 ± 0.51	59.07 ± 2.79	33.39 ± 1.29	53.54 ± 2.26	70.4 ± 3.09	37.25 ± 1.43
Gemma-2b	IT	41.9 ± 1.99	41.82 ± 3.29	28.75 ± 1.36	47.52 ± 3.45	66.47 ± 0.99	24.94 ± 8.68
		40.72 ± 0.93	37.17 ± 4.46	27.94 ± 1.26	45.69 ± 2.89	63.42 ± 1.35	29.41 ± 0.16
	PCDST-IT	43.15 ± 2.48	40.57 ± 5.31	28.66 ± 1.49	46.38 ± 1.63	67.03 ± 3.51	33.1 ± 1.49
		43.03 ± 2.1	40.85 ± 3.95	28.43 ± 1.5	45.82 ± 1.22	67.66 ± 4.07	32.39 ± 1.35
Gemma2-9b	IT	52.47 ± 0.59	62.69 ± 1.92	33.92 ± 2.09	65.97 ± 1.3	66.67 ± 2.23	33.1 ± 0.66
		51.26 ± 1.63	61.57 ± 2.45	33.44 ± 0.91	65.61 ± 1.71	65.97 ± 1.89	29.73 ± 6.27
	PCDST-IT	53.08 ± 2.74	65.63 ± 1.66	34.62 ± 1.95	62.96 ± 3.71	66.35 ± 4.23	35.84 ± 4.04
		52.46 ± 2.67	65.09 ± 1.17	33.92 ± 1.47	61.31 ± 3.52	65.97 ± 4.17	36.01 ± 3.98

We ran each experiment three times with different random seeds, and we report the mean and standard deviation here. The results with a gray background are the test results on the Instruction Test_A, while those with a white background are the test results on the Instruction Test_B.

Appendix D. The Similarity between the Slot Value Distributions of Two Slots

We added a similarity measure based on the cosine similarity between the slot value distributions of two slots in their respective domain validation sets. Figure A1 shows the heatmap of slot value distribution similarity between any two slots in MultiWOZ. Each cell in the figure represents the value of slot value distribution cosine similarity between the slots labeled in the horizontal axis and the vertical axis. The higher the brightness, the higher the similarity. The figure is symmetric because the cosine similarity is symmetric.

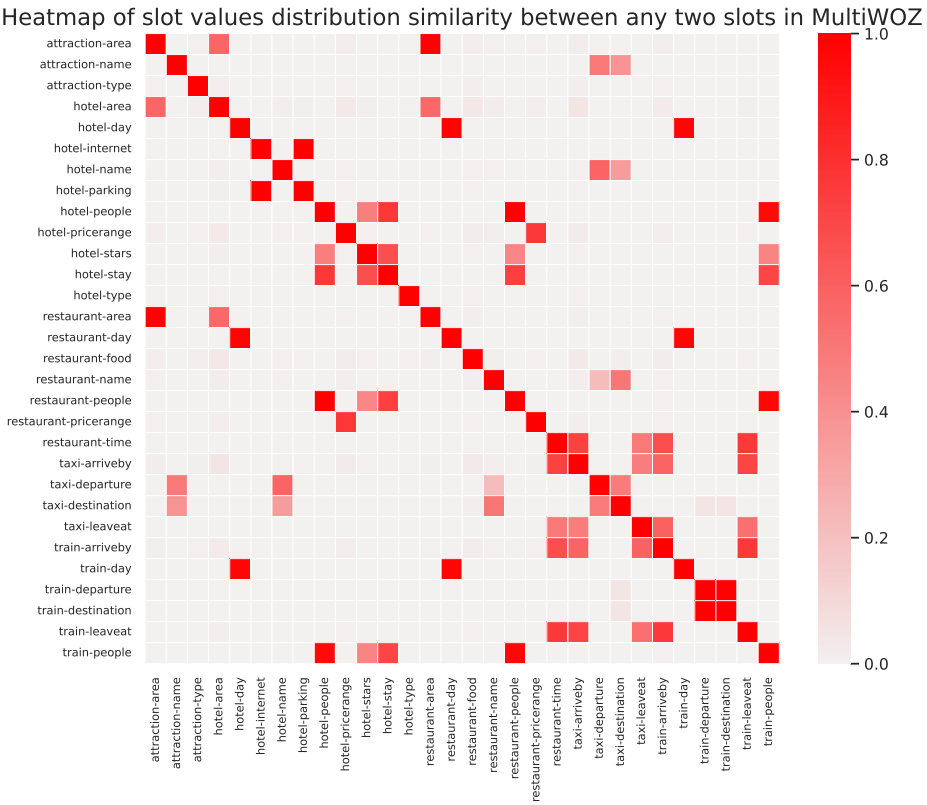


Figure A1. Heatmap of slot value distribution similarity between any two slots in MultiWOZ.

References

1. Huang, Z.; Li, F.; Yao, J.; Chen, Z. Mgcrl: Multi-view graph convolution and multi-agent reinforcement learning for dialogue state tracking. *Neural Comput. Appl.* **2024**, *36*, 4829–4846. [CrossRef]
2. Lee, Y.; Kim, T.; Yoon, H.; Kang, P.; Bang, J.; Kim, M. Dstea: Improving dialogue state tracking via entity adaptive pre-training. *Knowl.-Based Syst.* **2024**, *290*, 111542. [CrossRef]
3. Liu, Y.; Chen, L.; Yu, K. Label-aware auxiliary learning for dialogue state tracking. In Proceedings of the ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Seoul, Republic of Korea, 14–19 April 2024; pp. 11641–11645.
4. Jia, X.; Zhang, R.; Peng, M. Multi-domain gate and interactive dual attention for multi-domain dialogue state tracking. *Knowl.-Based Syst.* **2024**, *286*, 111383. [CrossRef]
5. Xu, J.; Song, D.; Liu, C.; Hui, S.C.; Li, F.; Ju, Q.; He, X.; Xie, J. Dialogue state distillation network with inter-slot contrastive learning for dialogue state tracking. *Proc. AAAI Conf. Artif. Intell.* **2023**, *37*, 13834–13842. [CrossRef]
6. Du, M.; Cheng, L.; Xu, B.; Wang, Z.; Wang, S.; Yuan, J.; Pan, C. Ts-dst: A two-stage framework for schema-guided dialogue state tracking with selected dialogue history. In Proceedings of the 2022 International Joint Conference on Neural Networks (IJCNN), Padua, Italy, 18–23 July 2022; pp. 1–8.
7. Tian, X.; Huang, L.; Lin, Y.; Bao, S.; He, H.; Yang, Y.; Wu, H.; Wang, F.; Sun, S. Amendable generation for dialogue state tracking. In Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI, Online, November 2021; pp. 80–92.
8. Heck, M.; van Niekerk, C.; Lubis, N.; Geishauser, C.; Lin, H.-C.; Moresi, M.; Gasic, M. Trippy: A triple copy strategy for value independent neural dialog state tracking. In Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue, Online, 1–3 July 2020; pp. 35–44.
9. 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, Florence, Italy, 28 July–2 August 2019; pp. 808–819.
10. Lee, H.; Lee, J.; Kim, T.-Y. Sumbt: Slot-utterance matching for universal and scalable belief tracking. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, Florence, Italy, 28 July–2 August 2019; pp. 5478–5483.
11. Kumar, A.; Ku, P.; Goyal, A.; Metallinou, A.; Hakkani-Tur, D. Ma-dst: Multi-attention-based scalable dialog state tracking. *Proc. AAAI Conf. Artif. Intell.* **2020**, *34*, 8107–8114. [CrossRef]
12. Li, S.; Cao, J.; Sridhar, M.; Zhu, H.; Li, S.-W.; Hamza, W.; McAuley, J. Zero-shot generalization in dialog state tracking through generative question answering. In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, Online, 19–23 April 2021; pp. 1063–1074.

13. Wang, Q.; Cao, Y.; Li, P.; Fu, Y.; Lin, Z.; Guo, L. Slot dependency modeling for zero-shot cross-domain dialogue state tracking. In Proceedings of the 29th International Conference on Computational Linguistics, Gyeongju, Republic of Korea, 12–17 October 2022; pp. 510–520.
14. Han, T.; Huang, C.; Peng, W. Coreference augmentation for multi-domain task-oriented dialogue state tracking. *arXiv* **2021**, arXiv:2106.08723.
15. Lin, Z.; Liu, B.; Moon, S.; Crook, P.A.; Zhou, Z.; Wang, Z.; Yu, Z.; Madotto, A.; Cho, E.; Subba, R. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. In Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Online, 6–11 June 2021; pp. 5640–5648.
16. Aksu, I.T.; Kan, M.-Y.; Chen, N. Prompter: Zero-shot adaptive prefixes for dialogue state tracking domain adaptation. *arXiv* **2023**, arXiv:2306.04724 .
17. Xu, H.-D.; Mao, X.-L.;Yang, P.; Sun, F.; Huang, H. Cross-domain coreference modeling in dialogue state tracking with prompt learning. *Knowl.-Based Syst.* **2024**, *283*, 111189. [CrossRef]
18. Chen, L.; Lv, B.; Wang, C.; Zhu, S.; Tan, B.; Yu, K. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. *Proc. Aaai Conf. Artif. Intell.* **2020**, *34*, 7521–7528. [CrossRef]
19. Hosseini-Asl, E.; McCann, B.; Wu, C.-S.; Yavuz, S.; Socher, R. A simple language model for task-oriented dialogue. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 20179–20191.
20. Victor, S.; Albert, W.; Colin, R.; Stephen, B.; Lintang, S.; Zaid, A.; Antoine, C.; Arnaud, S.; Arun, R.; Manan, D.; et al. Multitask prompted training enables zero-shot task generalization. In Proceedings of the International Conference on Learning Representations, Virtual Event, 25–29 April 2022.
21. Wang, Y.; Mishra, S.; Alipoormolabashi, P.; Kordi, Y.; Mirzaei, A.; Naik, A.; Ashok, A.; Dhanasekaran, A.S.; Arunkumar, A.; Stap, D.; et al. Super-naturalinstructions: Generalization via declarative instructions on 1600+ nlp tasks. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, Abu Dhabi, United Arab Emirates, 7–11 December 2022; pp. 5085–5109.
22. Wei, J.; Bosma, M.; Zhao, V.; Guu, K.; Yu, A.W.; Lester, B.; Du, N.; Dai, A.M.; Le, Q.V. Finetuned language models are zero-shot learners. In Proceedings of the International Conference on Learning Representations, Vienna, Austria, 3–7 May 2021.
23. Gao, S.; Agarwal, S.; Jin, D.; Chung, T.; Hakkani-Tur, D. From machine reading comprehension to dialogue state tracking: Bridging the gap. In Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI, Online, 9 July 2020; pp. 79–89.
24. Heck, M.; Lubis, N.; Ruppik, B.; Vukovic, R.; Feng, S.; Geishauser, C.; Lin, H.-C.; van Niekerk, C.; Gasic, M. Chatgpt for zero-shot dialogue state tracking: A solution or an opportunity? In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers), Toronto, ON, Canada, 9–14 July 2023; pp. 936–950.
25. Hu, Y.; Lee, C.-H.; Xie, T.; Yu, T.; Smith, N.A.; Ostendorf, M. In-context learning for few-shot dialogue state tracking. *arXiv* **2022**, arXiv:2203.08568.
26. Yang, Y.; Li, Y.; Quan, X. Ubar: Towards fully end-to-end task-oriented dialog system with gpt-2. *Proc. AAAI Conf. Artif. Intell.* **2021**, *35*, 14230–14238. [CrossRef]
27. Feng, Y.; Wang, Y.; Li, H. A sequence-to-sequence approach to dialogue state tracking. 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), Virtual Event, 1–6 August 2021; pp. 1714–1725.
28. Lee, C.-H.; Cheng, H.; Ostendorf, M. Dialogue state tracking with a language model using schema-driven prompting. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 4937–4949.
29. Su, Y.; Shu, L.; Mansimov, E.; Gupta, A.; Cai, D.; Lai, Y.-A.; Zhang, Y. Multi-task pre-training for plug-and-play task-oriented dialogue system. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), Dublin, Ireland, 22–27 May 2022; pp. 4661–4676.
30. Mi, F.; Wang, Y.; Li, Y. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. *Proc. AAAI Conf. Artif. Intell.* **2022**, *36*, 11076–11084. [CrossRef]
31. Budzianowski, P.; Wen, T.-H.; Tseng, B.-H.; Casanueva, I.; Ultes, S.; Ramadan, O.; Gasic, M. Multiwoz-a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv*, **2018**, arXiv:1810.00278.
32. Eric, M.; Goel, R.; Paul, S.; Sethi, A.; Agarwal, S.; Gao, S.; Kumar, A.; Goyal, A.; Ku, P.; Hakkani- Tur, D. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In Proceedings of the 12th Language Resources and Evaluation Conference, Marseille, France, 11–16 May 2020; pp. 422–428.
33. Veron, M.; Galibert, O.; Bernard, G.; Rosset, S. Attention modulation for zero-shot cross-domain dialogue state tracking. In Proceedings of the 3rd Workshop on Computational Approaches to Discourse, Gyeongju, Republic of Korea, 16–17 October 2022; pp. 86–91.
34. Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; Sutskever, I. Language models are unsupervised multitask learners. *OpenAI blog* **2019**, *1*, 9.
35. Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; Liu, P.J. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.* **2020**, *21*, 1–67.

36. Lin, Z.; Liu, B.; Madotto, A.; Moon, S.; Zhou, Z.; Crook, P.A.; Wang, Z.; Yu, Z.; Cho, E.; Subba, R.; et al. Zero-shot dialogue state tracking via cross-task transfer. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, Punta Cana, Dominican Republic, 7–11 November 2021; pp. 7890–7900.
37. Loshchilov, I.; Hutter, F. Decoupled weight decay regularization. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
38. Touvron, H.; Lavril, T.; Izacard, G.; Martinet, X.; Lachaux, M.-A.; Lacroix, T.; Rozière, B.; Goyal, N.; Hambro, E.; Azhar, F.; et al. Llama: Open and efficient foundation language models. *arXiv* **2023**, arXiv:2302.13971.
39. Touvron, H.; Martin, L.; Stone, K.; Albert, P.; Almahairi, A.; Babaei, Y.; Bashlykov, N.; Batra, S.; Bhargava, P.; Bhosale, S.; et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv* **2023**, arXiv:2307.09288.
40. Dubey A.; Jauhri A.; Pandey A.; et al. The llama 3 herd of models. *arXiv* **2024**, arXiv:2407.21783.
41. Gemma Team; Mesnard, T.; Hardin, C.; Dadashi, R.; Bhupatiraju, S.; Pathak, S.; Sifre, L.; Rivière, M.; Kale, M.S.; Love, J.; et al. Gemma: Open models based on gemini research and technology. *arXiv* **2024**, arXiv:2403.08295.
42. Gemma Team; Riviere M.; Pathak S., et al. Gemma 2: Improving Open Language Models at a Practical Size. *arXiv* **2024**, arXiv:2408.00118.