Prompt Learning for Few-Shot Dialogue State Tracking

Yuting Yang[1,2], **Wenqiang Lei**[3], **Pei Huang**[2,4], **Juan Cao**[1,2], **Jintao Li**[1] **and Tat-Seng Chua**[5]
[1]Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
[3]Sichuan University
[4]State Key Laboratory of Computer Science, Institute of Software, Chinese Academy of Sciences
[5]National University of Singapore
yangyuting@ict.ac.cn, wenqianglei@gmail.com, huangpei@ios.ac.cn
caojuan@ict.ac.cn, jtli@ict.ac.cn, dcscts@nus.edu.sg

Abstract

Collecting dialogue state labels, slots and values, for learning dialogue state tracking (DST) models can be costly, especially with the wide application of dialogue systems in new-rising domains. In this paper, we focus on how to learn a DST model efficiently with limited labeled data. We design a prompt learning framework for few-shot DST, which consists of two main components: value-based prompt and inverse prompt mechanism. This framework aims to utilize the language understanding and generation ability of pre-trained language models (PLM). First, we design value-based prompt functions to probe the DST-related knowledge from PLM, which do not rely on the known ontology of slots. Further, an inverse prompt mechanism is utilized to self-check the "prompted" knowledge and help the PLM understand the essence of DST task further. Experiments show that our model can generate unseen slots and outperforms existing state-of-the-art few-shot methods. It indicates that DST-related knowledge can be probed from PLM and utilized to address low-resource DST efficiently with the help of prompt learning.

1 Introduction

Dialogue state tracking (DST) modules, which aim to extract dialogue states during conversation (Young et al., 2013), is an important component for task-oriented dialog systems to understand users' goals and needs (Wen et al., 2017; Lei et al., 2018). Dialogue states are sets of slots and their corresponding values. Collecting state labels can be costly (Budzianowski et al., 2018), requiring experts to indicate all *(slot, value)* information for each turn in dialogues.

To reduce the dependency on large amounts of training data, some few-shot methods are proposed recently for low-resource DST. Most of them apply domain transfer-based methods (Wu et al., 2019; Lee and Jha, 2019; Rastogi et al., 2020a) which rely on the assumption about the similarity among different domains. Another series of approaches has tried to exploit external knowledge. Chen et al. (2013) and Hudecek et al. (2021) consider slots and frames are similar semantic units and use the FrameNet semantic parsers to automatically induce slots. Wu et al. (2020a) fine-tune BERT with a task-oriented dialogue dataset and utilize it for the downstream DST task. Recently, a new paradigm, "*Pre-train, Prompt and Predict*" (Liu et al., 2021a), which aims at utilizing PLM in a more effective way, has aroused the public's attention. It is supposed to be useful in few-shot scenarios as it can "probe" the task-related knowledge from PLM efficiently using a prompt. In 2021, Lee et al. (2021) introduce prompt learning for DST task. This work aims to encode slot information into PLM via prompt and improve prediction performance when there are plenty of labeled data. However, the potential of prompt learning for low-resource DST is still under-explored. To further exploit this paradigm, we design a prompt learning frame-

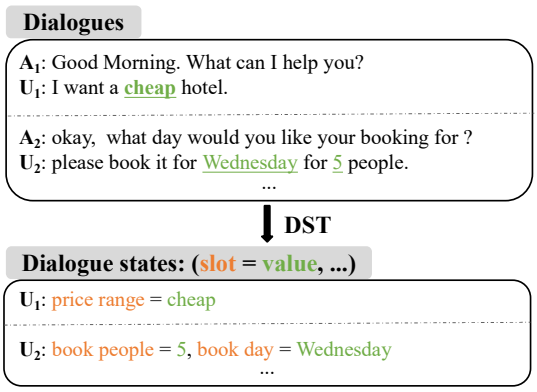

Figure 1: Dialogue state tracking (DST) task. U and A represent user's and system's utterances respectively. DST aims to extract dialogue states pairs (slot, value), for each user's utterance. Values are usually the explicit needs expressed in the utterances.

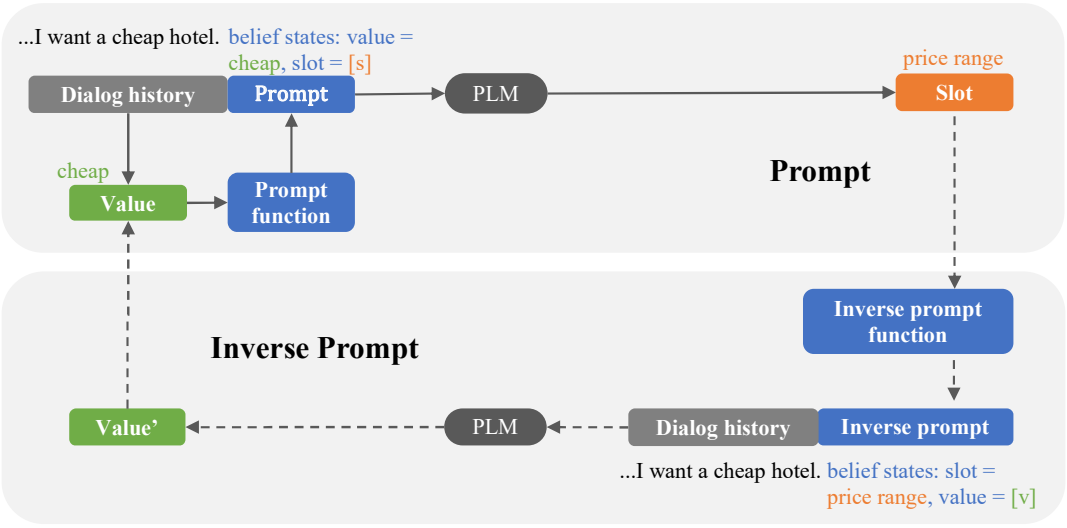

Figure 2: Overview of the prompt-learning framework for few-shot DST. While training, prompt function $f(v)$ accepts value candidate v and get the **value-based prompt**, which is concatenated to dialog history for PLM to generate the corresponding slot s. Dashed arrows demonstrate the process of **inverse prompt** mechanism which gets inverse prompt via the inverse prompt function $I(s)$ for the generated s and aims to generate value v' which is supposed to be close to the original input value v.

work for low-resource DST, which has value-based prompt module and inverse prompt module.

First, we design **value-based prompt functions** which use values as inputs to construct prompts. Slots are the target outputs. For few-shot scenarios, the slots that appear in the few labeled dataset may not include all possible needs a user will propose. Defining all possible slots in advance is also difficult because of the rapid application in different domains and users' continuous needs. %% Thus, we rethink about DST task and consider values as prompts to probe the corresponding slots. We design value-based prompt functions which equip the textual prompt with values. As shown in Figure 2, a prompt function is a textual template, e.g., "*belief states: value = [v], slot = [s]*". Given the dialogue history "*...Plan a train to London*", after extracting the value candidate *London*, the prompt becomes "*belief states: value = London, slot = [s]*" where *[s]* is supposed to be generated as *destination* by the PLM. Value-based prompting method can alleviate the dependency on pre-defined slot types and generate unseen slots.

In addition, values and slots are all semantic units that describe the users' needs during conversations. Prompting values via slots (value generation) can be seen as a dual-task of prompting slots with values (slot generation). Thus, we design **inverse prompt** mechanism as shown in Figure 2. While training, after generating slots s via value-based prompts, slots are presented to the inverse prompt function I. The inverse prompt process aims to generate the corresponding value v' which is supposed to be close to the original input v. Naturally, there exists an internal correlation between these two types of prompt tasks and they can benefit each other, especially under the few shot settings. The inverse prompt mechanism can also help to self-check and restrict the output of the prompt process: if a generated slot can be used to prompt the original value, the value belongs to the slot with a larger probability. Finally, a simple but effective ensemble method is used to leverage the complementary advantages of different prompt functions while testing.

The main contributions of our work can be summarized as (1) We reformulate DST as a language modeling task for slots generation and design value-based prompt functions to probe DST-related knowledge from PLM for low-resource scenarios. The framework doesn't rely on the known ontology of target slots. (2) We propose an inverse prompt mechanism to further help the PLM understand the essence of DST with few labels and tune the generated slots. (3) Experimental results show that our model can generate unseen slots and significantly outperforms state-of-the-art few-shot approaches.

2 Preliminary

2.1 Prompt Learning

Prompt learning, which aims to utilize pre-trained language models more effectively with the help of *prompt*, is a new NLP paradigm ("*Pre-train, Prompt and Predict*") proposed recently. Usually, the original task input x is used to construct a prompt that can reformulate the original task into a language modeling task. Take the emotion classification task as an example, when recognizing the emotion of a social media post, *"I missed the bus today"*, we may continue with a prompt *"I felt so __"*, and ask the PLM to fill the blank with an emotion-bearing word. With the appropriate prompts, PLM can be pushed to generate the task-related output directly.

Given the *prompt function* f which maps original input x to the prompt, the goal is to learn:

$$P\left(y \mid x, f(x)\right) \tag{1}$$

where y is the *answer* to be generated/filled. In DST, y can be a word in dialogue state B.

2.2 Dialogue State Tracking

We consider each conversation with T-turn utterances alternating between the user and system: $c = \{a_1, u_1, ...a_T, u_T\}$ where u_t and a_t represent the user's and system's utterance respectively. Given the dialog history c_t (including current user utterance u_t and the former utterances $h_t = \{a_1, u_1, ...u_{t-1}, a_t\}$), a DST model aims to extract the dialogue state (belief state) B_t for u_t which comprises multiple tuples of slots s and their associated values v ($B_t = \{(s_1, v_1), ...(s_n, v_n)\}$). For example, given the dialog history c_t ("*...Plan a train to London on this Tuesday*"), DST model is supposed to generate belief states $B_t = \{(destination, London), (day, this\ Tuesday)\}$). The goal is to learn probability distribution P (Peng et al., 2021) for t-th turn:

$$P\left(B_t \mid c_t\right) \tag{2}$$

If B_t is considered as a word sequence (Hosseini-Asl et al., 2020), DST is essentially a language modeling task. Large-scale pre-trained language models (PLM) show outstanding language modeling and generation ability. Following the existing paradigm (*Pre-train and Fine-tune*), we need to fine-tune PLM with the task-related dataset. Fine-tuning with few labeled dataset may lead to over-fitting. Thus, an effective way to help PLM understand DST task in their familiar way (language modeling) and utilize the generation ability is important, inspiring the exploration of prompt learning for few-shot DST.

3 Prompt Learning for Few-Shot DST

In the following section, we will describe the main components of the designed prompt learning framework for few-shot DST, including **value-based prompt** (Section 3.1), **inverse prompt** (Section 3.2) and **prompt ensemble** (Section 3.3).

3.1 Value-based Prompt

Previous work (Jiang et al., 2020; Schick and Schütze, 2021; Cui et al., 2021) show that the design of prompting function f is a key factor that influences the final performance. Usually, the function function contains a *template* which is a textual string and remains two blanks to be filled (the input and the answer).

A natural idea is using slots as prompts to generate corresponding values, which is called slot-based prompt (Lee et al., 2021). For example, given c ("*...plan a train to London.*") and slot (*destination*), the input of PLM becomes "*...Plan a train to London. destination [y]*" where *[y]* is supposed to be generated as *"London"*. This method relies on the known ontology of slot type. For few-shot DST, the slots that appear in the few labeled datasets may not include all possible needs. In addition, defining all possible slots are difficult as the rapid application in different new-rising domains and user's continuous need. In the real-world application, the candidate set of s may be unknown and changeable.

Thus, we rethink about DST task and consider values as prompts to generate the corresponding slots as shown in Figure 2. Such a method doesn't require any knowledge about the target slot types and thus can generate unseen slots under the generative framework. Four intuitive templates for value-based prompts are shown in Table 1. Take the first template for an example (*"belief states: value = [v], slot = [s]"*), given value candidate v = *"London"*, $f(v)$ = *"belief states: value = London, slot = [s]"*. The goal is to learn the probability of slots in t-th turn given c_t and the value v:

$$P\left(s \mid c_t, f(v)\right) \tag{3}$$

The overall learning objective of this generation processing is maximizing the log-likelihood of

	Prompt Functions
f_1	belief states: value = [v], slot = **[s]**
f_2	belief states: [v] = **[s]**
f_3	[v] is the value of **[s]**
f_4	What is the slot type of [v] ? **[s]**

Table 1: Different prompt functions f. [v] is the input of value candidate and [s] is the slot to be generated.

slots in the training dataset D:

$$\mathcal{L} = \sum_{t}^{|D|} log\ P\left(s_t \mid c_t, f(v_t)\right) \tag{4}$$

As a turn may contain multiple values and slots, each pair of (slot, value) constructs an instance for training and testing. While training, the values are known with annotations. Real values are utilized to construct prompts and train the slot-generation process. While testing, values are unknown. Referring to (Goel et al., 2019; Min et al., 2020), candidate values are extracted from current user's utterance. We extract POS tags (Cui and Zhang, 2019) and consider adjective and adverb as possible values as well as their previous negator (e.g., "*not expensive*"). Named entities (e.g., name of place, time) and co-references are extracted via Stanford CoreNLP toolkit (Manning et al., 2014). Stop words and repeated candidates are filtered. Each value candidate is used to construct the value-based prompt and generate the slot. Actually, value generation can also be modeled as a summarization task and take advantage of few-shot summarization methods, which can be improved in future work but is not the focus of this paper.

3.2 Inverse Prompt

Values and slots are both core semantic units in utterances that describe users' needs. Thus, prompting values with slots (value generation) can be seen as a dual-task of prompting slots with values (slot generation). Naturally, these two types of prompt tasks are supposed to hold an intrinsic correlation correlation and can benefit each other, especially in the few-shot settings.

Thus, we design *inverse prompt* as shown in Figure 2. While training, after generating the slot (s) via value-based prompts, s is presented to inverse prompt function (I). The inverse prompt process aims to answer the corresponding value v' which is supposed be close to the original input one v. We take "*belief states: [s] = [v]*" as the template in I. The loss function $\widetilde{\mathcal{L}}$ for inverse prompt mechanism is:

$$\widetilde{\mathcal{L}} = \sum_{t}^{|D|} log\ P\left(v_t \mid c_t, I(s_t)\right) \tag{5}$$

The final loss function $\mathcal{L}^*$ consists of loss functions in prompt learning $\mathcal{L}$ and inverse prompt learning $\widetilde{\mathcal{L}}$:

$$\mathcal{L}^* = \mathcal{L} + w * \widetilde{\mathcal{L}} \tag{6}$$

where w is a decimal in $(0, 1)$ and used to adjust the influence of inverse prompt.

There are two main differences between our work and existing work using slots as prompts: (1) Inverse prompt is used as an auxiliary task for the original prompt learning. Our goal is to utilize it to help PLM understand the task and tune the output further: if a slot can be used to prompt the original value, it means there is a larger probability that the value belongs to the generated slot. (2) In our work, the slot types are not static. We generate the slots in the whole vocabulary space, making generating unseen slots is possible.

3.3 Prompt Ensemble

In the previous section, we described methods to generate a set of value-based prompt functions as shown in Table 1. Each of these prompts may be more or less effective at eliciting knowledge from the PLM, and thus it is necessary to decide how to use these generated prompts at test time. Unfortunately, under few-shot settings, it's hard to get enough training and development set to automatically select or generate the best-performing prompt (Jiang et al., 2020; Gao et al., 2021; Ben-David et al., 2021; Davison et al., 2019; Liu et al., 2021b). We introduce a multi-prompt learning method (*prompt ensemble*) for few-shot DST task in this section to effectively utilize different prompts.

Prompt ensemble methods use multiple unanswered prompts for input at inference time to make predictions (Liu et al., 2021a). It can leverage the complementary advantages of different prompts and alleviate the cost of choosing one best-performing prompt. There is relatively little work on prompt ensemble for generation tasks. A simple way for ensemble in this case it to train a separate model for each prompt and generate the output based on the vocabulary distribution learned by several models while testing. The probability of slot

s_t is calculated via:

$$P\left(s_t \mid c_t\right)=\sum_{k}^{K} \alpha_k * P\left(s_t \mid c_t, f_k(v_t)\right) \quad (7)$$

where f_k is the k-th prompt and α_k is its weight. K is the number of prompt functions.

4 Experiments

4.1 Experimental Setup

Datesets We evaluate our methods on MultiWOZ 2.1 (Eric et al., 2019) dataset. It's a multi-domain task-oriented dialog dataset and contains 8438/1000/1000 dialogues for training/validation/testing, respectively. Following existing work (Wu et al., 2019), we keep five domains (*Restaurant, Hotel, Attraction, Taxi, Train*) because the other two domains only appear in the training set. Each turn can include multiple slots.

Evaluation Metrics We adopt the standard metric in DST (Wu et al., 2019): joint goal accuracy (JGA). The metric compares the entire predicted belief states to the gold one at each dialog turn. The prediction is considered correct if and only if all the predicted states exactly match the ground truth states. Only when the values and slots are both correct, the prediction is correct. To omit the influence of the effect of value extraction, we also present the corresponding accuracy (JGA*) while the values are correctly identified for our methods.

Implement Details We choose SOLOIST (Peng et al., 2021) as our base model. SOLOIST is initialized with the 12-layer GPT-2 (Radford et al., 2019) and further trained on multiple task-oriented dialog corpora (Schema (Rastogi et al., 2020b) and Taskmaster (Byrne et al., 2019)) for two dialogue-related tasks (belief prediction and response generation). Specifically, the belief prediction task accepts utterance as input and generates the belief states as a word sequence (e.g., "*Belief state: destination = London*"). Thus, we suppose that knowledge about DST may be learned via SOLOIST, and what we need to do is to find an effective way to "probe" the knowledge and apply it to few-shot scenarios. In addition, the moderate size of SOLOIST (117M parameters) makes fine-tuning for the task-related prompts computationally efficient. α for each prompt function in Eq.7 is set to the same value (1/4). w in Eq.6 is 0.1. Inverse prompt is used for training. While testing, value-based prompts are given to the models for target slot generation.

Baselines We compare our methods with several strong baselines capable of few-shot inference, which achieve SoTA on MultiWOZ 2.1 dataset. (1) **TRADE** (Wu et al., 2019) requires the embedding of slots as inputs and uses a soft copy mechanism to either copy the corresponding values from utterance pairs or generate them using RNN. (2) **Self-Sup** (Wu et al., 2020b) adds two self-supervised objectives: preserving latent consistency and modeling conversational behavior for TRADE. (3) **TOD-BERT** (Wu et al., 2020a) trained BERT with several task-oriented dialogue relevant tasks: masked language modeling and response generation with large-scale corpora (100k dialogues across over 60 different domains). For DST, it learns a classifier to predict the value over the pre-defined possible value set for each known slot. (4) **DSI** (Min et al., 2020) introduces latent variable models for DST which consider dialogue states and slots as latent variables. Although the original paper didn't focus on few-shot scenarios, we compare it here as both of us use the same ways of extracting value candidates as inputs and aim to generate slots. (5) **SOLOIST**. It's the base model of our method. To investigate the effect of prompt learning, we fine-tune SOLOIST on the DST task which accepts dialog history and uses values as prompts directly to generate slots.

It's worthy to mention that except SOLOIST, most baselines need pre-defined slots while inference and some need the entire ontology including both slots and their all possible value sets. Although DSI doesn't need pre-defined slot types, the number of target slots should be given.

4.2 Experiment Results

To simulate the few-shot scenarios, we randomly select a limited quantity of labeled training data for training. Similar to the previous few-shot works (Wu et al., 2020a,b), we conduct experiments on the data ratio of 1%, 5%, 10% and 25%. Specifically, to observe the performance in an extreme low-resource scenario, we also compare their performances while the training ratio is only 0.1% (8 dialogues and less than 60 users' utterances).

Few-shot performances are shown in Table 2. We can observe that our model outperforms existing works by a large margin. From 0.1% to 10%, our model increases JGA by more than 7%, indi-

	0.1%	1%	5%	10%	25%
	JGA				
TRADE	1.4	10.4	27.7	32.6	38.5
Self-Sup	15.4	19.5	30.6	34.5	40.2
TOD-BERT	6.3	9.9	28.6	37.0	44.3
DSI	1.0	1.1	1.3	1.3	2.6
SOLOIST	26.8	36.4	37.1	37.9	39.4
Ours	**36.1**	**44.3**	**44.7**	**44.7**	**45.4**
	JGA*				
Ours	**75.4**	**92.4**	**94.7**	**94.7**	**95.1**

Table 2: Comparison of different models under the different ratios of training data (0.1% corresponds to less than 60 users' utterances).

Dialogue history: ...[user]i need a taxi to pick me up at regency gallery and take me to Don Pasquale Pizzeria. **Gold value:** Don Pasquale Pizzeria **Our value:** (**Missed**)
Dialogue history: ...[user] i am looking for city centre north bed and breakfast **Gold value:** city centre north bed and breakfast **Our value:** north (**Partially extracted**)
Dialogue history: ...[system] any preference on star rating [user] no , that s not important to me . **Gold value:** don't care **Our value:** not important (**Totally wrong**)

Table 3: Error analyses for extracting value candidates. Three main types of errors are listed (missed, partially extracted and totally wrong).

cating the superiority of our model in low-resource scenarios. For 25%, the improvement is relatively small (1%). It may be owing to that using 1% data almost performs the same as using 25% data (1.2% difference) for our model. Using a few labeled data can help the PLM understand DST task.

We then analyze the results of value extracting and find it can only achieve an accuracy less than 61%. Table 3 lists the three main types of wrong extraction: missed, partially extracted and totally wrong. However, our model still outperforms others as JGA in Table 2 shows. It attributes to the high accuracy of slot generation while turn-level values are correctly extracted. When the ratio of training data is 1%, 92.4% turn-level states are correctly generated given correct value candidates as JGA* shown in Table 2. The high values of JGA* indicate the potential of performance along with the improvement of value extraction in future work.

4.3 Unseen Slot Generation

For MultiWOZ2.1, we present the slots of each domain in Table 4. We find that some domains share some slots with other domains. For example, all slots of *Attraction* can be found in *Hotel*. On the contrary, some domains hold some slots that

slots
$area^{123}$, $arrive\ by^{45}$, day^{235}, $departure^{45}$, $destination^{45}$, $food^{3}$, $internet^{2}$, $leave^{45}$, $name^{123}$, $people^{235}$, $parking^{2}$, $price^{23}$, $stars^{2}$, $stay^{2}$, $time^{3}$, $type^{12}$

Table 4: All slots in MultiWOZ2.1. The upper script on slot indicates the domain it belongs to (1: *Attraction*, 2: *Hotel*, 3: *Restaurant*, 4: *Taxi*, 5: *Train*).

are not seen in other domain. For *Hotel*, it has four unseen slots: *parking*, *book stay*, *stars* and *internet*. *Restaurant* has two unseen slots (*food* and *time*). Here, we consider "unseen slots" as both "unseen" in the labeled training data and "unseen" in the slot names of the source domains.

To observe the extension and generation ability for unseen slots, we design two zero-shot experiments: leave *Hotel* or *Restaurant* as held-out-domain respectively, and train on other four domains. We present slots accuracy which evaluates the slot-level accuracy of correctly generated slots while values are correctly extracted. From the results in Figure 3(a), we find that:

(1) For seen slots that have the same names as that of source domains, our model can generate them with high accuracy. For example, *area* in *Hotel* domain is a common slot for other two source domains (*Attraction* and *Restaurant*), which can be generated with 96.92% accuracy. It indicates the good transfer ability across domains.

(2) For some unseen slots (*book stay* and *stars* in *Hotel* of Figure 3(a), *book time* and *food* in *Restaurant* of Figure 3(b)), our model can generate them with more than 87% accuracy. For example, given the dialogue history "*...yes, please book it for 1 person and for 5 nights starting Friday.*" The model successfully generates "*book stay*" for "*5*" even it has never seen the instances of book stay while training. Without known slot types, our model can infer the hidden semantic from the value and contexts, which is supposed to be the slot.

(3) Figure 3(a) misses two unseen slots (*parking* and *internet*). For these two slots, the value extraction module fails to extract its gold value ("*yes*") from utterance. The accuracy of value extraction is 0. So, the slot accuracy is also 0. However, we extract the adjective "*free*" from the user utterance as shown in Table 5. Then PLM model can infer that the word "*free*" is the property of "*internet*". Actually, we think "*internet: free*" can also clearly describe a user's need. The case indicates a possible drawback of the existing annotation system.

(a) *Hotel* (b) *Restaurant*

Figure 3: Slot accuracy of each slot in *Hotel* and *Restaurant* domain under zero-shot settings. X-axis is the slot accuracy and y-axis is the slot. Pink bars mark unseen slots.

Dialogue history: ... [system] could you please give me any preferences for internet and parking ? [user] I would like it to be a guesthouse that has free WiFi.
Gold states: yes, internet
Our states: free, internet

Table 5: A test instance that has a wrong value candidate ("free") and a correct generated slot ("internet"). However, we think (free, internet) can also describe the user's need here.

	f_1	f_2	f_3	f_4	En
Orig	24.3	30.9	25.7	27.7	31.9
+ Inv	32.4	31.4	29.6	33.6	**36.1**

Table 6: JGA results for our models trained with 0.1% data. "+Inv" means adding inverse prompt mechanism for the original models ("orig") given different prompt functions (from f_1 to f_4). "En" means the ensemble of models trained on different prompt functions with and without inverse prompt.

4.4 Ablation Studies

We further observe the performances of different components including different value-based prompt functions, inverse prompt mechanism and prompt ensemble. We first train separate models with each value-based prompt ("Orig" for $f_1, ...f_4$), then add inverse prompt mechanism for each of them ("+Inv"). Finally, we apply prompt ensemble ("En") for the trained models with and without inverse prompt respectively. Experiments with 0.1% training data are shown in Table 6.

We observe that: (1) The first four numerals in the first row show the original performance with different prompt functions (no inverse prompt mechanism and prompt ensemble). Among the four prompts, f_2 performs best without inverse prompt which may attribute to the similar format of f_2 compared with the output sequences in a pre-training task of SOLOIST (Considering dialogue history as inputs and generate dialogue states in the format as "*belief states: [s1] = [v1], [s2] = [v2]*"). It's also the reason why inverse prompt doesn't improve its performance much as the prompt ("*belief states: [v] = [s]*") and inverse prompt ("*belief states: [s] = [v]*") are too similar to learn complementary knowledge. (2) Inverse prompt brings improvement for all prompt functions, especially for the "weakest" prompt function f_1. (3) The prompt ensemble enables further improvement for both models with and without inverse prompt. Under few-shot settings, prompt ensemble is a simple but efficient way of utilizing different prompt functions.

4.5 Analyses about the Weight of Inverse Prompt

We conduct experiments to observe the influence of weight w in Eq.6. w is set to 0, 0.1, 0.3, 0.5. Experiments using 0.1% training data and different value-based prompts are shown in Figure 4. We find that the JGA performance always increases with the value of w first and then begins to decrease. It means that the inverse prompt is actually an auxiliary task and can provide useful knowledge when the weight is relatively small. All experiments for the four prompts perform best when the w is 0.1. So we set it to 0.1 in the former experiments.

5 Related Work

5.1 Few-Shot Dialogue State Tracking

Some few-shot methods used data augmentation to get more labeled data for training. Campagna

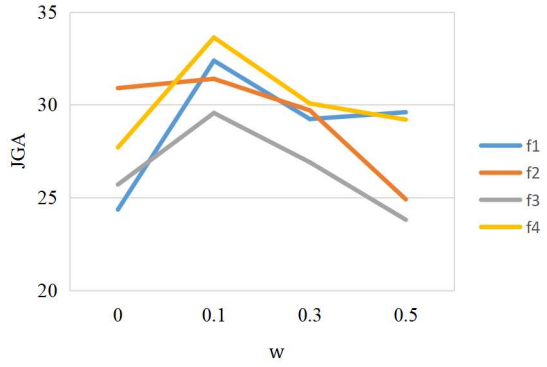

Figure 4: The influence of weight w for inverse prompt using different prompt functions f. Experiments with $w = 0.1$ always perform best for all prompt functions.

et al. (2020) and Hou et al. (2021) propose to synthesize dialogues for a new domain using the small number of domain templates derived from observing a small dataset and the ontology of the domain. These methods depend on the ontology about slots of the target domain.

Most of the existing work focuses on transferring from other resource-rich DST domains. Lee and Jha (2019) and Rastogi et al. (2020a) utilize the slot description for transferring reusable concepts across domains. Wu et al. (2020a) learn similarity functions between slots and values, and transfer them into unseen domains. Dingliwal et al. (2021) introduces meta-learning and uses source domains to meta-learn the parameters of the model used to initialize the fine-tuning process of the target domain. One constraint of such methods is that they rely on domain similarity for transfer, and therefore cannot be applied to general domains.

Another thread of approaches tries to exploit external knowledge. Chen et al. (2013) and Hudecek et al. (2021) utilize FrameNet-style (Fillmore et al., 1976) semantic frames and named entity recognition (NER) as the weak supervision for slot candidates. Gao et al. (2019),Gao et al. (2020), Li et al. (2021) and Lin et al. (2021) reformulate DST into a Reading Comprehension (RC) task and make use of the abundant RC data and frameworks to overcome the data scarcity issue in the DST task. Wu et al. (2020b) investigate two self-supervised objectives: preserving latent consistency and modeling conversational behavior. However, they have limited performance owing to the limited common knowledge.

5.2 Prompt Learning

With the rapid development of large-scale pre-trained language models (PLM), a new paradigm arise public's attention: "*pre-train, prompt, and predict* (Liu et al., 2021a)". Instead of adapting PLM to downstream tasks via objective engineering, prompt learning reformulates downstream tasks to look more like those solved during the original PLM training with the help of a textual prompt. GPT-3 model (Brown et al., 2020) achieves remarkable few-shot performance solely by leveraging a few task demonstrations as input context (e.g., *"Translate English into French"*) and a natural-language prompt (e.g., *"cheese ==> "*). However, training such a huge model (175B parameters) is difficult. A more usual prompt learning method is "prompt-based fine-tune": utilize a moderately-sized PLM for which fine-tuning is computationally efficient and fine-tune it with the task-related prompts. It shows good performance in many few-shot scenarios. Gao et al. (2021) use RoBERT-large and design automatic prompt generation for text classification. Li and Liang (2021) add continuous task-specific vector as prompt to each transformer layer and achieve improvements in low-resource text summarization. For DST task, Lee et al. (2021) use slots as prompt directly and generate the corresponding values, which needs a lot of labeled training data for fine-tuning PLM. For few-shot DST, the prompt learning-based methods are still under-explored.

6 Conclusion

For the lack of labeled data in practical DST tasks, we design a prompt learning framework, which consists of two main components (value-based prompt and inverse prompt mechanism). Our model can effectively probe DST-related knowledge from pre-trained language models and utilize it for DST task. Experiments show that our model outperforms existing state-of-the-art methods under different levels of resources. It achieves an improvement of 7.3% joint goal accuracy under the extreme low-resource settings (only 0.1% training data). In addition, this framework doesn't rely on the known ontology of slot types. With extensive experiments, we find that it can generate slots that are not seen in source domains and are not pre-defined as well with high probabilities. In the future, we'll focus on improving the performance of extracting value candidates.

References

Eyal Ben-David, Nadav Oved, and Roi Reichart. 2021. Pada: A prompt-based autoregressive approach for adaptation to unseen domains. *arXiv preprint arXiv:2102.12206*.

Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, pages 5016–5026. Association for Computational Linguistics.

Bill Byrne, Karthik Krishnamoorthi, Chinnadhurai Sankar, Arvind Neelakantan, Ben Goodrich, Daniel Duckworth, Semih Yavuz, Amit Dubey, Kyu-Young Kim, and Andy Cedilnik. 2019. Taskmaster-1: Toward a realistic and diverse dialog dataset. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 4515–4524. Association for Computational Linguistics.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S. Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 122–132. Association for Computational Linguistics.

Yun-Nung Chen, William Yang Wang, and Alexander I. Rudnicky. 2013. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *2013 IEEE Workshop on Automatic Speech Recognition and Understanding, Olomouc, Czech Republic, December 8-12, 2013*, pages 120–125. IEEE.

Leyang Cui, Yu Wu, Jian Liu, Sen Yang, and Yue Zhang. 2021. Template-based named entity recognition using BART. In *Findings of the Association for Computational Linguistics: ACL/IJCNLP 2021, Online Event, August 1-6, 2021*, volume ACL/IJCNLP 2021 of *Findings of ACL*, pages 1835–1845. Association for Computational Linguistics.

Leyang Cui and Yue Zhang. 2019. Hierarchically-refined label attention network for sequence labeling. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 4113–4126. Association for Computational Linguistics.

Joe Davison, Joshua Feldman, and Alexander M. Rush. 2019. Commonsense knowledge mining from pretrained 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 2019, Hong Kong, China, November 3-7, 2019*, pages 1173–1178. Association for Computational Linguistics.

Saket Dingliwal, Shuyang Gao, Sanchit Agarwal, Chien-Wei Lin, Tagyoung Chung, and Dilek Hakkani-Tür. 2021. Few shot dialogue state tracking using meta-learning. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 1730–1739. Association for Computational Linguistics.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *CoRR*, abs/1907.01669.

Charles J Fillmore et al. 1976. Frame semantics and the nature of language. In *Annals of the New York Academy of Sciences: Conference on the origin and development of language and speech*, volume 280, pages 20–32. New York.

Shuyang Gao, Sanchit Agarwal, Tagyoung Chung, Di Jin, and Dilek Hakkani-Tür. 2020. From machine reading comprehension to dialogue state tracking: Bridging the gap. *CoRR*, abs/2004.05827.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tür. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue, SIGdial 2019, Stockholm, Sweden, September 11-13, 2019*, pages 264–273. Association for Computational Linguistics.

Tianyu Gao, Adam Fisch, and Danqi Chen. 2021. 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, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 3816–3830. Association for Computational Linguistics.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. In *Interspeech 2019, 20th Annual Conference of the International Speech Communication Association, Graz, Austria, 15-19 September 2019*, pages 1458–1462. ISCA.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*.

Yutai Hou, Sanyuan Chen, Wanxiang Che, Cheng Chen, and Ting Liu. 2021. C2c-genda: Cluster-to-cluster generation for data augmentation of slot filling. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, AAAI 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, IAAI 2021, The Eleventh Symposium on Educational Advances in Artificial Intelligence, EAAI 2021, Virtual Event, February 2-9, 2021*, pages 13027–13035. AAAI Press.

Vojtech Hudecek, Ondrej Dusek, and Zhou Yu. 2021. Discovering dialogue slots with weak supervision. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2430–2442. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know. *Trans. Assoc. Comput. Linguistics*, 8:423–438.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. *CoRR*, abs/2109.07506.

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *The Thirty-Third AAAI Conference on Artificial Intelligence, AAAI 2019, The Thirty-First Innovative Applications of Artificial Intelligence Conference, IAAI 2019, The Ninth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2019, Honolulu, Hawaii, USA, January 27 - February 1, 2019*, pages 6642–6649. AAAI Press.

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1437–1447.

Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, and Julian J. McAuley. 2021. 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, EACL 2021, Online, April 19 - 23, 2021*, pages 1063–1074. Association for Computational Linguistics.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 4582–4597. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Paul A. Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, and Pascale Fung. 2021. Zero-shot dialogue state tracking via cross-task transfer. *CoRR*, abs/2109.04655.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2021a. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *CoRR*, abs/2107.13586.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021b. GPT understands, too. *CoRR*, abs/2103.10385.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Rose Finkel, Steven Bethard, and David McClosky. 2014. The stanford corenlp natural language processing toolkit. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics, ACL 2014, June 22-27, 2014, Baltimore, MD, USA, System Demonstrations*, pages 55–60. The Association for Computer Linguistics.

Qingkai Min, Libo Qin, Zhiyang Teng, Xiao Liu, and Yue Zhang. 2020. Dialogue state induction using neural latent variable models. In *Proceedings of the Twenty-Ninth International Joint Conference on Artificial Intelligence, IJCAI 2020*, pages 3845–3852. ijcai.org.

Baolin Peng, Chunyuan Li, Jinchao Li, Shahin Shayandeh, Lars Liden, and Jianfeng Gao. 2021. SOLOIST: building task bots at scale with transfer learning and machine teaching. *Trans. Assoc. Comput. Linguistics*, 9:907–824.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020a. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8689–8696. AAAI Press.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8689–8696. AAAI Press.

Timo Schick and Hinrich Schütze. 2021. 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, EACL 2021, Online, April 19 - 23, 2021*, pages 255–269. Association for Computational Linguistics.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina Maria Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve J. Young. 2017. 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, EACL 2017, Valencia, Spain, April 3-7, 2017, Volume 1: Long Papers*, pages 438–449. Association for Computational Linguistics.

Chien-Sheng Wu, Steven C. H. Hoi, Richard Socher, and Caiming Xiong. 2020a. TOD-BERT: pre-trained natural language understanding for task-oriented dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 917–929. Association for Computational Linguistics.

Chien-Sheng Wu, Steven C. H. Hoi, and Caiming Xiong. 2020b. Improving limited labeled dialogue state tracking with self-supervision. In *Findings of the Association for Computational Linguistics: EMNLP 2020, Online Event, 16-20 November 2020*, volume EMNLP 2020 of *Findings of ACL*, pages 4462–4472. Association for Computational Linguistics.

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Conference of the Association for Computational Linguistics, ACL 2019, Florence, Italy, July 28- August 2, 2019, Volume 1: Long Papers*, pages 808–819. Association for Computational Linguistics.

Steve J. Young, Milica Gasic, Blaise Thomson, and Jason D. Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proc. IEEE*, 101(5):1160–1179.

A Example Appendix

This is an appendix.