Chongqing University of Technology

A Dual Prompt Learning Framework for Few-Shot Dialogue State Tracking

Yuting Yang
Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences; University of Chinese Academy of Sciences
Beijing, China
yangyuting@ict.ac.cn

Wenqiang Lei
Sichuan University
Sichuan, China
wenqianglei@gmail.com

Pei Huang
Stanford University
California, USA
huangpei@stanford.edu

Juan Cao
Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences; University of Chinese Academy of Sciences
Beijing, China
caojuan@ict.ac.cn

Jintao Li
Key Lab of Intelligent Information Processing, Institute of Computing Technology, Chinese Academy of Sciences
Beijing, China
jtli@ict.ac.cn

Tat-Seng Chua
National University of Singapore
Singapore
dcscts@nus.edu.sg

https://github.com/YANG-Yuting/DPL

—— WWW 2023

Reported by Yuyang Lai

1.Introduction

2.Method

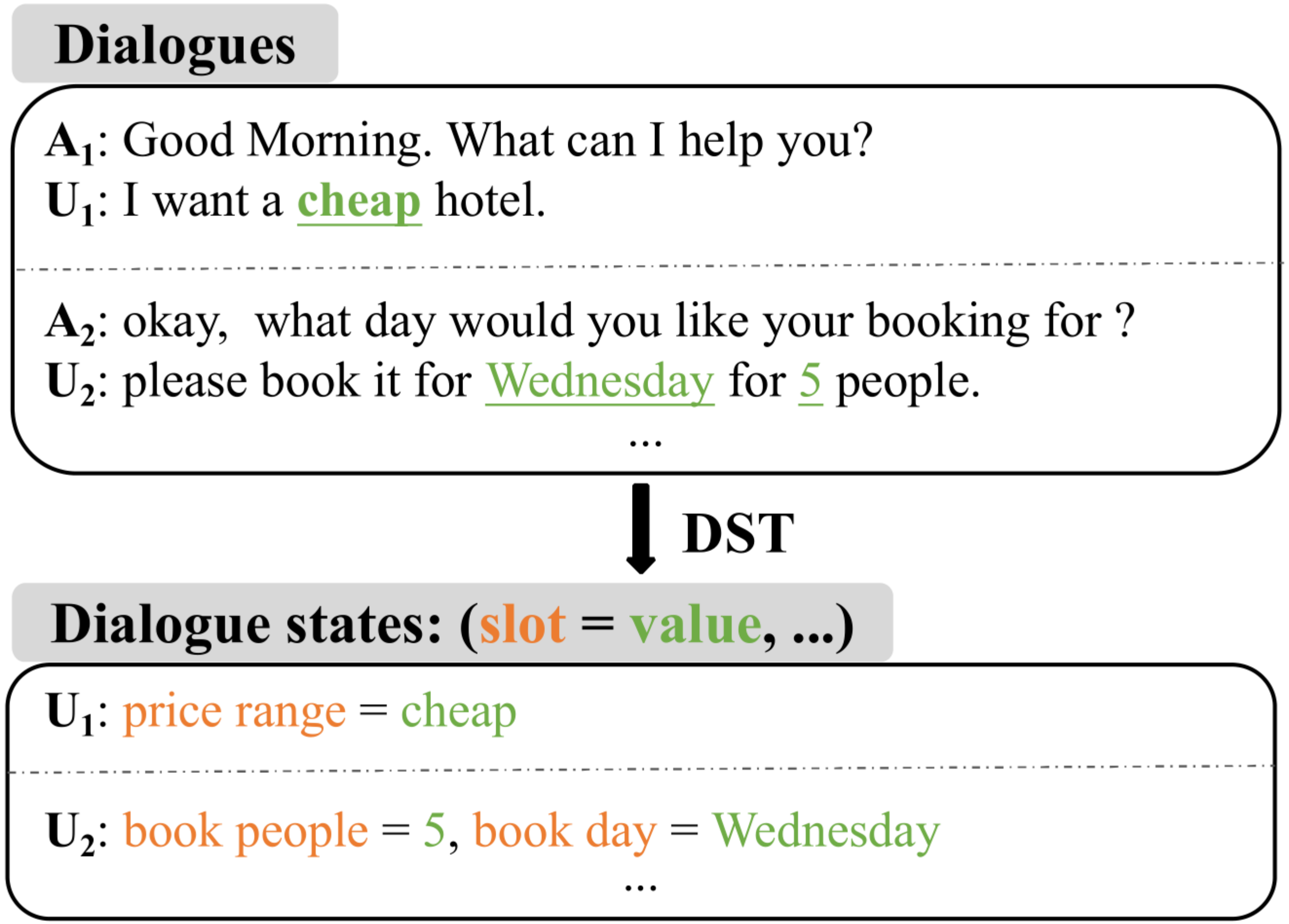
3.Experiments

Introduction

DST aims to extract dialogue states pairs (slot,value),for each user's utterance.

A slot describes an attribute about the user's need.

value is the value of the given attribute.



Introduction

Prompt Learning, which aims to utilize pre-trained language models more effectively with the help of prompt, is a new NLP paradigm.

I love this movie → I love this movie. Overall, it was a [Z] movie

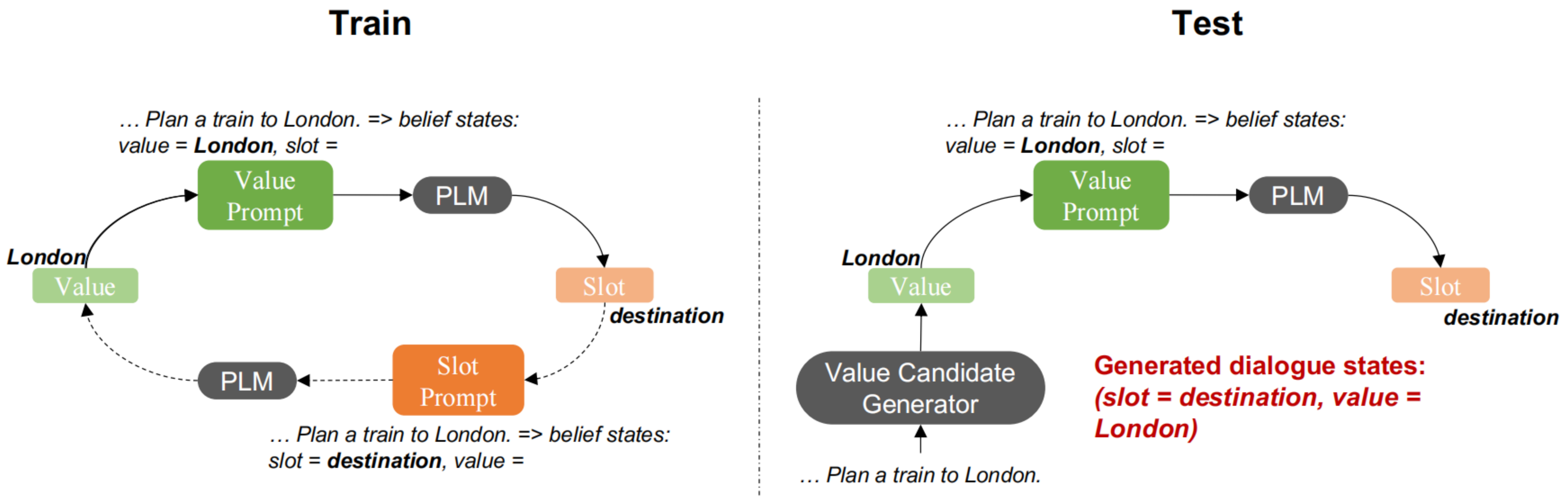


Figure 2: Overview of the dual prompt learning framework for few-shot DST. While training, two dual prompts are constructed: value prompt and slot prompt. Value prompt is constructed with a value and given to the PLM to generate corresponding slots. Slot prompt is constructed with slots and used to generate values. While testing, value candidates are first generated by a pre-trained value candidate generator, and then used to construct value prompts and generate slots.

Method

$$P\left(y \mid f(x)\right) \qquad (1)$$

"I missed the bus today"; *"I felt so __"*;

$$P\left(B_t \mid c_t\right) \qquad (2)$$

$c_t = \{a_1, u_1, ..., a_t, u_t\}$

$(B_t = \{(s_1, v_1), ...(s_n, v_n)$

"...Plan a train to London on this Tuesday"

$B_t = \{(destination, London), (day, this\ Tues\text{-}day)\}$

Method

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

$$\mathcal{L}_v = -\sum_{i}^{|D|} log\, P\left(s_i \mid f(c_i, v_i)\right) \tag{4}$$

$f(c, v)$ = "[c] belief states: value = London, slot = [s]"

$$\mathcal{L}_s = -\sum_{i}^{|D|} log\, P\left(v_i \mid I(c_i, s_i)\right) \tag{5}$$

"[c] belief states: [s] = [v]"

$$\mathcal{L} = \mathcal{L}_v + w * \mathcal{L}_s \tag{6}$$

Method

$$\mathcal{L}_r = -\sum_{i}^{|D|} R(v_i) * log\ P\ (v_i) \tag{7}$$

$$\mathcal{L}'' = \lambda\mathcal{L}' + (1-\lambda)\mathcal{L}_r \tag{8}$$

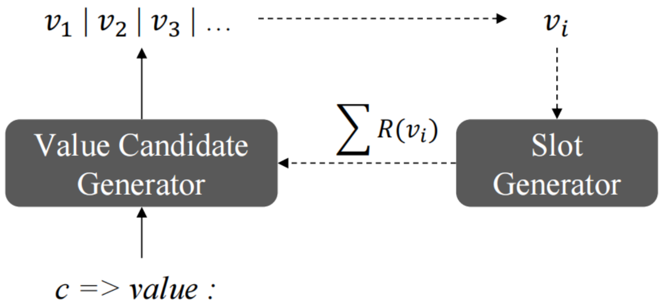


Figure 3: The process of value candidate generation. Dashed lines denote the tuning process: The slot generator accepts the generated value v_i to construct the value prompt and then feeds it into the fine-tuned PLM to generate slots and get reward for tuning the value candidate generator.

Method

$$P\left(s_i\right)=\sum_{k}^{K} \alpha_k * P\left(s_i \mid f_k(c_i, v_i)\right) \tag{9}$$

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

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

Experiments

	1%	5%	10%	20%	25%
Need slot ontology					
TRADE	9.7	29.4	34.1	N/A	41.4
Self-Sup	20.4	33.7	37.2	N/A	42.7
TOD-BERT	10.3	27.8	38.8	N/A	44.3
No need for slot ontology					
SimpleTOD	7.9	16.1	22.4	31.2	N/A
MinTL	9.3	21.3	30.3	36.0	N/A
SOLOIST	13.2	26.5	32.4	38.7	N/A
PPTOD	29.7	40.2	43.5	47.0	N/A
DPL	**33.7**	**42.1**	**45.6**	**49.5**	**51.2**

Table 2: Few-shot experimental results on MultiWOZ 2.0.

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 3: All slots in MultiWOZ 2.1. The upper script on slot indicates the domain it belongs to (1: *Attraction*, 2: *Hotel*, 3: *Restaurant*, 4: *Taxi*, 5: *Train*).

Experiments

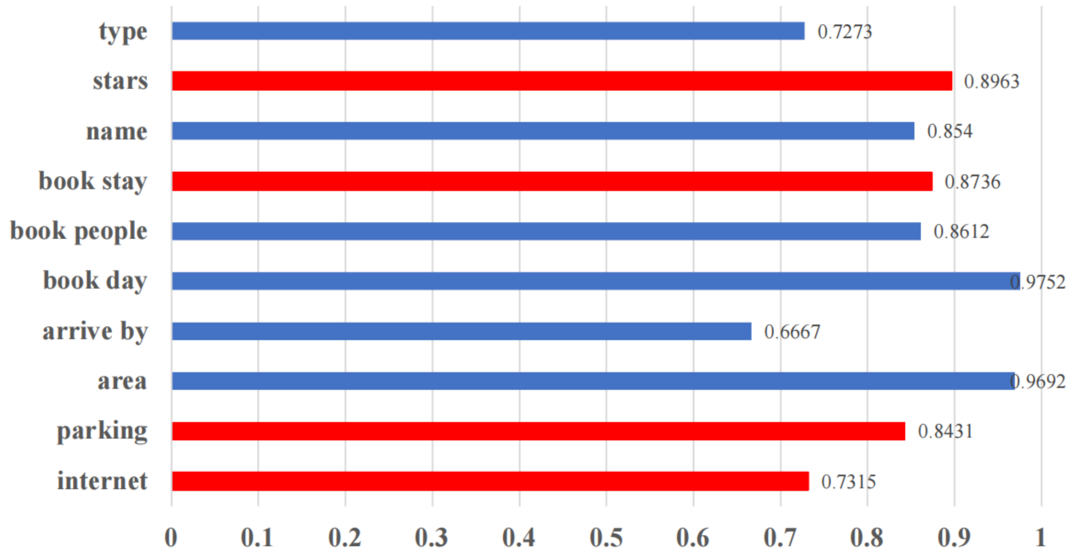


Figure 4: Slot accuracy of each slot in Hotel domain under zero-shot settings. X-axis is the slot accuracy and y-axis is the slot. Red bars mark unseen slots.

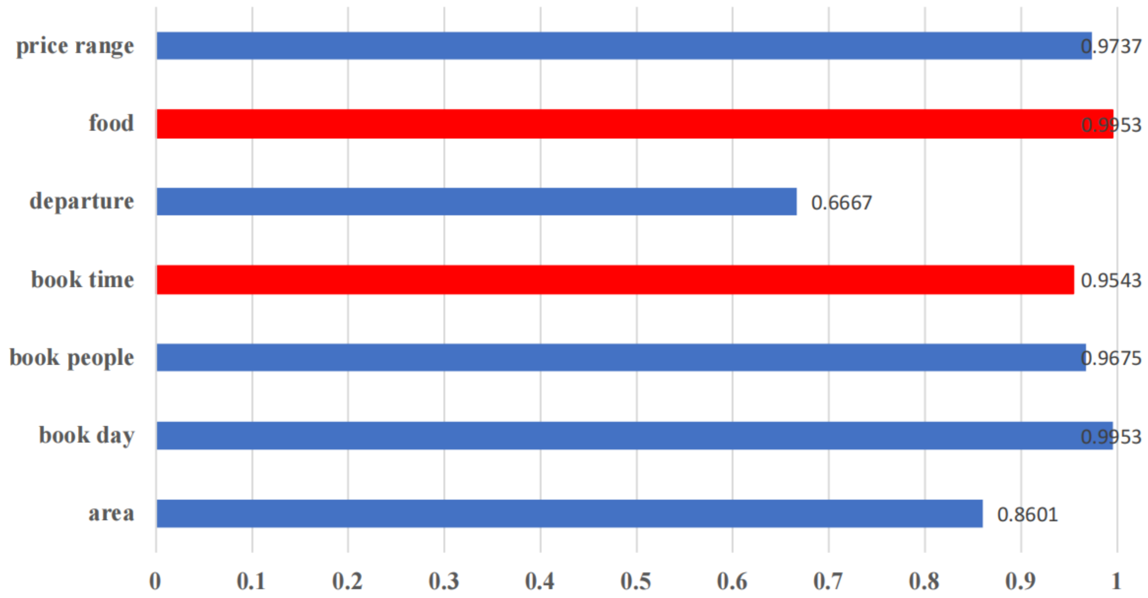


Figure 5: Slot accuracy of each slot in Restaurant.

Experiments

Model	Attraction			Hotel			Restaurant			Taxi			Train		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
TRADE	35.8	57.5	63.1	19.7	37.4	41.4	42.4	55.7	60.9	63.8	66.5	70.1	59.8	69.2	71.1
DSTQA	N/A	70.5	71.6	N/A	50.2	53.7	N/A	59.0	64.5	N/A	70.9	74.2	N/A	70.4	74.5
T5DST	58.8	65.7	69.5	43.1	50.7	54.9	57.6	61.9	63.5	70.1	73.7	74.7	70.8	74.2	77.6
DPL	**60.4**	**70.5**	**72.1**	**45.7**	**53.1**	**56.9**	**60.5**	**64.3**	**67.2**	**74.1**	**76.4**	**77.8**	**72.1**	**76.3**	**79.0**

Table 4: Few-shot cross-domain experimental results on MultiWOZ 2.0.

Experiments

rule	32.65			
	1%	5%	10%	25%
Ours	51.42	59.22	63.11	65.17
Ours *w/o tuning*	47.58	55.93	61.57	65.03

Table 5: Turn-level accuracy on test set of value generator under different ratios of training data. "*w/o tuning*" means removing the process of using the output of slot generation to tune the process of value generation.

	f_1	f_2	f_3	f_4	*En*
DPL	25.7	29.4	26.4	28.9	33.7
DPL *w/o slot prompt*	20.1	29.1	22.3	24.5	29.5

Table 7: JGA results for our models trained with 1% data given different prompt functions (from f_1 to f_4). "*w/o slot prompt*" means removing the training process of slot prompt. "En" shows the result of the ensemble of models trained on different prompt functions with and without slot prompt.

Dialogue history: ... [user] no , i do not care where it is . i like 3 stars and i absolutely need free wifi .
Gold values: don't care, 3, yes
Generated values: don't care, 3, yes

Table 6: A test instance whose values are generated by the trained value generator with 25% training data. It shows that the value generator can generate implicit values ("yes").

Experiments

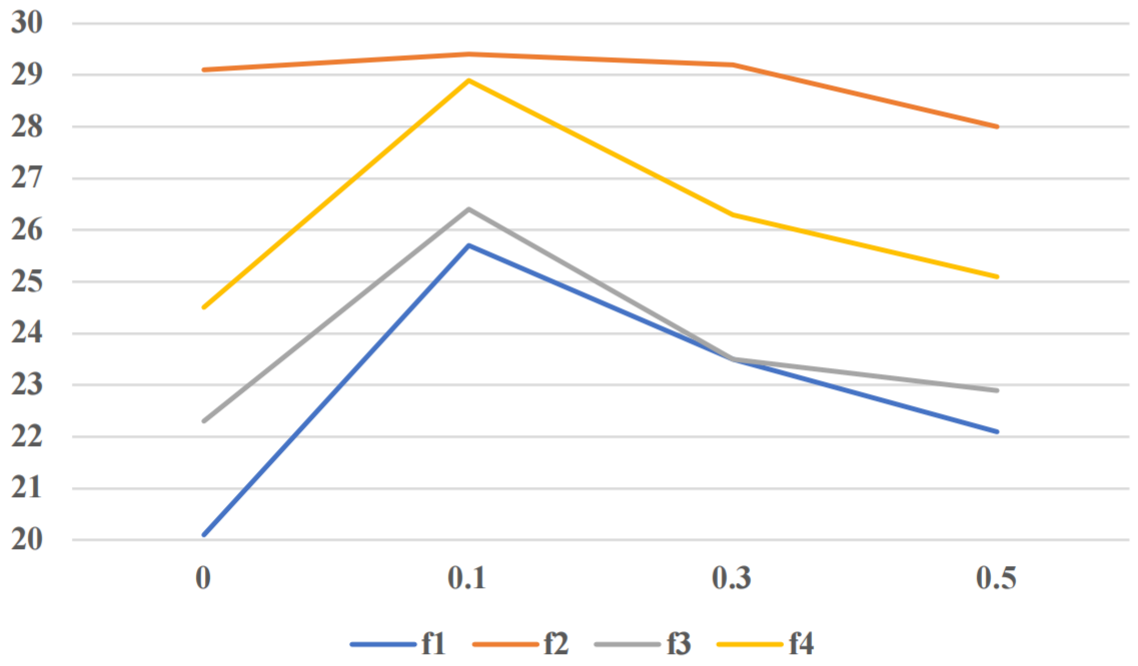


Figure 6: The influence of weight w for slot prompt using different prompt functions f. X-axis is the value of w and y-axis is JGA. Experiments with $w = 0.1$ always perform best for all prompt functions.

Thank you!