CINS: Comprehensive Instruction for Few-shot Learning in Task-oriented Dialog Systems

Fei Mi[†] **Yitong Li**[‡] **Yasheng Wang**[†]
[†]**Huawei Noah's Ark Lab** [‡]**Huawei Technologies Co., Ltd.**
{mifei2,liyitong3,wangyasheng}@huawei.com

Abstract

As the labeling cost for different modules in task-oriented dialog (ToD) systems is high, a major challenge is to learn different tasks with the least amount of labeled data. Recently, pre-trained language models (PLMs) have shown promising results for few-shot learning in ToD. To better utilize the power of PLMs, this paper proposes Comprehensive Instruction (CINS) that exploits PLMs with extra task-specific instructions. We design a schema *(definition, constraint, prompt)* of instructions and their customized realizations for three important downstream tasks in ToD, ie. intent classification, dialog state tracking, and natural language generation. A sequence-to-sequence model (T5) is adopted to solve these three tasks in a unified framework. Extensive experiments are conducted on these ToD tasks in realistic few-shot learning scenarios with small validation data. Empirical results demonstrate that the proposed CINS approach *consistently* improves techniques that finetune PLMs with raw input or short prompt.

1 Introduction

Large-scale pre-trained language models (PLMs), such as BERT (Devlin et al. 2019), UniLM (Dong et al. 2019), GPT (Radford et al. 2018), GPT-2 (Radford et al. 2019), T5 (Raffel et al. 2020) and GPT-3 (Brown et al. 2020), have shown tremendous success in various NLP applications, especially in few-shot or zero-shot learning scenarios. In task-oriented dialog (ToD) systems, the labeling cost is very high such that the size of well-labeled data is often small. Therefore, few-shot learning in ToD is especially important and valuable in many practical applications.

Many attempts have been proposed to leverage PLMs to improve few-shot learning in ToD. For example, Chen, Zhuo, and Wang (2019); Chao and Lane (2019); Kale and Rastogi (2020b) directly finetune a PLM on downstream ToD tasks. However, the general objectives and tasks during the model pre-training phase are often very different from the formulation of specific downstream ToD tasks. To bridge this gap, Kale and Rastogi (2020a); Lin et al. (2021) propose to slightly transform the input of downstream ToD tasks to better-matched tasks that PLMs have seen during pre-training. Such perspective is similar to a recent line of methods, called **Prompting** or **Prompt Engineering** (Schick and Schütze 2021a,b; Gao, Fisch, and Chen 2021; Schick and Schütze 2020; Liu et al. 2021b), to better exploit the capabilities of PLMs.

In Prompting, the input is modified using a "template" to form a "prompt" to feed to a PLM. By defining new templates, it unifies the task, objective, and formulation between downstream tasks and pre-training, and Prompting shows strong performance in several few-shot or even zero-shot learning scenarios. However, one limitation of Prompting is that the "prompts" are often *short and concise* (Schick and Schütze 2021b; Gao, Fisch, and Chen 2021). We conjecture that the massive amount of information stored in large PLMs might not be adequately exploited with short prompts only. Therefore, we are going to study: *can extra instructions further improves short prompts to exploit the few-shot capability of PLMs for ToD?*

To this end, we propose "**Comprehensive Instruction** (CINS)". Besides a short and concise prompt, we additionally include task-specific *Definition* and *Constraint*. *Task Definition* provides a high-level natural language definition and nature of the task itself. *Task Constraint* additionally gives fine-grained task-specific constraint w.r.t output space (e.g. candidate labels, label descriptions, etc.) generated by the PLM. We formulated the overall schema and task-specific realization of CINS for three downstream tasks (intent classification, dialog state tracking, and natural language generation) in ToD. Furthermore, we adopt a Seq2Seq PLM (T5) as a unified framework to solve these three tasks. In our experiments, we adopt a "*realistic few-shot learning*" setting that only uses small validation data with the same size as the few-shot training data. We contend that this is a more reasonable few-shot setting compared to existing few-shot ToD studies (Mi et al. 2019; Peng et al. 2020b; Kale and Rastogi 2020a; Lin et al. 2021) that use full-size validation data.

The main contribution of this paper is three-fold:

- This is the first attempt to systematically study the effect of add extra task-specific instructions to better exploit pre-trained models for ToD.
- We propose "Comprehensive Instruction (CINS)" with a unified schema and task-specific realizations for different ToD tasks. CINS serves as a complement of Prompting to better guide the behavior of powerful PLMs.

- We conduct extensive experiments on three ToD downstream tasks, including intent classification, dialog state tracking, and natural language generation. A realistic few-shot learning setting is adopted by only utilizing small validation data. Empirical results demonstrate that CINS consistently and notably improves state-of-the-art methods (with or without Prompting) in realistic few-shot learning scenarios.

2 Related Work

2.1 Prompting Pre-trained Language Models

PLMs have shown great success in a number of NLP applications, yet pre-training objectives are often different from downstream tasks. To bridge this gap. Prompting or Prompt Engineering (Liu et al. 2021b) has been recently studied. In this paper, we focus on "discrete prompting" where inputs are wrapped by discrete tokens. Explorations w.r.t "continuous prompting" (Li and Liang 2021; Liu et al. 2021c; Lester, Al-Rfou, and Constant 2021) or how to ensemble multiple prompts (Schick and Schütze 2021a,b; Jiang et al. 2020; Qin and Eisner 2021) are beyond the focus of this paper. An overview of related topics can be found at Liu et al. (2021b).

Discrete prompting (Schick and Schütze 2021a,b; Tam et al. 2021; Gao, Fisch, and Chen 2021; Schick and Schütze 2020; Liu et al. 2021a; Hu et al. 2021) transforms the input to a discrete textual string as a "prompt" to feed to a PLM. For classification tasks, a *verbalizer* is often used to map the PLM's output to task labels. The verbalizer can also be learned for classification tasks (Gao, Fisch, and Chen 2021; Hu et al. 2021). By defining templates with human intelligence, the few-shot or even zero-shot power of PLMs can be better exploited. As "prompts" are often short and concise, Mishra et al. (2021) proposed to encode extra task-specific instructions to generalize to new tasks. Our paper is motivated by such idea, while our focus is to study more fine-grained formulations for ToD tasks in realistic few-shot learning settings.

2.2 Pre-trained Language Models for ToD

Several large-scale PLMs have been applied to ToD. GPT-2 is applied by Budzianowski and Vulic (2019); Mi et al. (2020) to train a response generation model. Ham et al. (2020); Hosseini-Asl et al. (2020); Peng et al. (2020a) proposed to train GPT-2 on different sub-tasks (dialog state tracking, dialog act prediction, and response generation) as a sequence prediction problem. BERT is recently applied to different classfication tasks of ToD by Wu et al. (2020); Cai et al. (2021); Mi et al. (2021). T5 is also recently applied to ToD by Lin et al. (2021) for dialog state tracking and by Kale and Rastogi (2020a,b) for natural language generation. As GPT-style auto-regressive models are not strong for language understanding tasks, and BERT-style models are not suitable for generation tasks, we adopt Seq2Seq style PLMs, such as T5 or BART (Lewis et al. 2019), as a unified framework to solve different ToD tasks.

Several studies have also confirmed that PLMs are good few-shot learners for ToD. Peng et al. (2020a) demonstrated few-shot end-to-end response generation from dialog contexts with GPT-2. Peng et al. (2020b) validated GPT-2 for few-shot natural language generation. Wu et al. (2020) showed the effectiveness of BERT for several few-shot classification tasks in ToD. The few-shot capability of T5 was validated for dialog state tracking (Lin et al. 2021) as well as natural language (Kale and Rastogi 2020a,b). The idea of Lin et al. (2021); Kale and Rastogi (2020a) bears similarity to Prompting as the input is transformed to better match the pre-trained knowledge of PLMs, and we compared these methods in our paper.

3 Methodology

In §3.1, we first explain the Seq2Seq (T5) model and how it can unify three downstream tasks (intent classification, dialog state tracking, natural language generation) in ToD. In §3.2, we explain the unified schema of instructions and how to customize them for different ToD tasks.

3.1 Seq2Seq (T5) for ToD Tasks

We first provide an overview of Seq2Seq model followed by its applications in three tasks in ToD. As recently proposed by Lewis et al. (2019); Raffel et al. (2020), sequence-to-sequence models unify a variety of natural language processing task, including both classification and generation tasks as:

$$y = \mathrm{Seq2Seq}(x) = \mathrm{Dec}(\mathrm{Enc}(x))\,, \tag{1}$$

where both x and y are token sequences. Input x first goes through a sequence encoder followed by another sequence decoder to produce the output y. The model is trained used the standard maximum likelihood objective, i.e. using teacher forcing (Williams and Zipser 1989) and a standard cross-entropy loss. This formulation unifies different ToD tasks, and we elaborate on them later.

There are four common tasks in the ToD pipeline: natural language understanding (NLU), dialog state tracking (DST), dialog management (DM), and natural language generation (NLG). DM in practice highly depends on business logic to determine suitable system actions. Therefore, we focus on NLU, DST, and NLG tasks in this paper.

Intent Classification (IC) Intent classification is an essential task of NLU in ToD. It predicts the intent label of the user's utterance. For example, the model needs to predict the intent of an utterance "*I want to book a 5-star hotel*" as "*book hotel*". For IC, the input x to T5 is a user utterance U_k, and the output y of T5 is the intent label.

Dialog State Tracking (DST) Given a dialog history, the task of DST is to predict the value of slots predefined by an ontology. Following Lin et al. (2021), the model predicts the value for each (domain, slot) pair. A dialog history at turn t is a set of alternating utterances between user (U) and system (S), denoted as $C_t = \{U_1, S_1, \ldots, S_{t-1}, U_t\}$. To predict the value of a slot i, the dialog history C_t is concatenated with the name of description s_i of slot i. Then the concatenated sequence $\{C_t, s_i\}$ is fed to the encoder of T5 for the decoder to generate the value v_i of this slot.

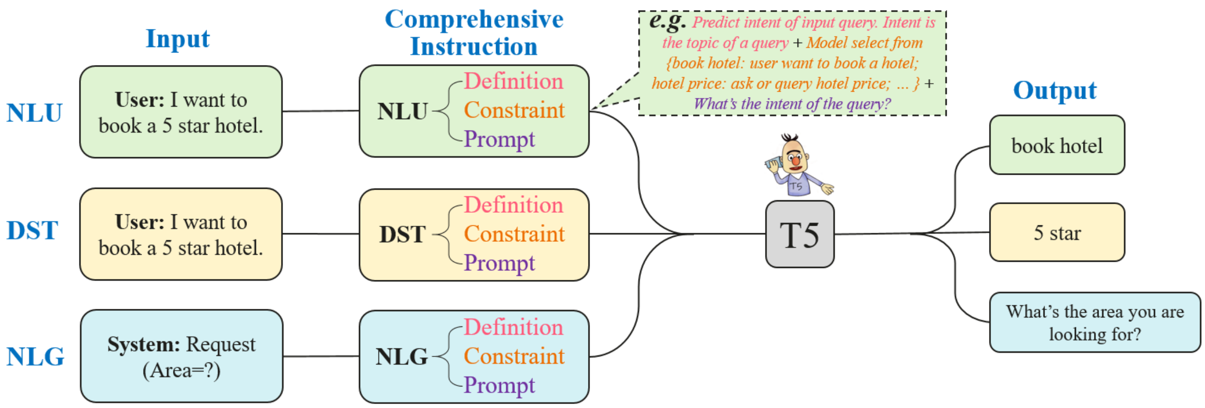

Figure 1: The unified framework of applying Comprehensive Instruction to different ToD downstream tasks. For each task in a row, the input is concatenated with the customized instruction (definition, constraint, prompt) before feeding to a T5 model to generate different types of output. An example instruction for NLU (intent classification) is given in the upper dashed box, and more details for different tasks are elaborated in Table 1.

Natural Language Generation (NLG) The natural language generation task is to produce a natural language utterance for a *semantic representation* called *dialog action* produced by the system. Following Peng et al. (2020b); Kale and Rastogi (2020b), the "**Naive**" canonical representation $\mathcal{A} = \sum_{i=1}^{A} a_i(s_i = v_i)$ is the set of actions produced by the system, where A is the total number of actions for this turn. Each action consists of a single intent a_i representing the semantics of the action, along with optional slot and value pairs $(s_i = v_i)$. For example,

$$\mathcal{A} = [\textit{Inform(name=Rosewood), Inform(star=5)}]\ .$$

However, $\mathcal{A}$ is different from the plain text seen in the pre-trained phase. To overcome the semantic limitation of $\mathcal{A}$, Kale and Rastogi (2020a) propose to first transform $\mathcal{A}$ to natural language $\mathcal{A}'$ using human-written templates. For example:

$$\mathcal{A}' = \textit{The hotel is called [Rosewood]. It is [5] star.}$$

The representation of $\mathcal{A}'$ is called Template Guided Text Generation (**"T2G2"**, Kale and Rastogi (2020a)), and it achieves strong few-shot NLG performance. The **Naive** representation $\mathcal{A}$ or **T2G2** $\mathcal{A}'$ is feed to the encoder of T5, and the decoder generates a natural language utterance as a response.

Examples of T5 for IC, DST, and NLG tasks are illustrated in Figure 1 without looking at the middle column ("Comprehensive Instruction").

3.2 Comprehensive Instruction for ToD

This section first explains two existing types (Standard, Prompt Engineering) of input to Seq2Seq. Then, we explain how to formulate the proposed method (Comprehensive Instruction), and how to design it for different downstream tasks in ToD.

Standard (STD) The standard input to Seq2Seq models is the raw set of input tokens. We explained different kinds of standard input to T5 in §3.1. For example, the raw token sequence *"I want to book a 5-star hotel"* serves as the input for intent classification to predict the label "book hotel".

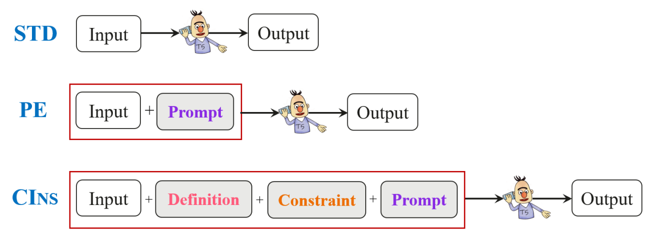

Figure 2: Formulation comparison for Standard input, Prompt Engineering, and Comprehensive Instruction.

Prompt Engineering (PE) To better utilize the capability of PLMs, PE constructs task-specific *prompts* around the raw input before feeding to PLMs. For example, *"'I want to book a 5-star hotel'* <u>*What does the previous query ask about?*</u>*"*. Underlined tokens in blue are human-designed prompt tokens to help the PLM understand the task. The idea of PE is to find a proper prompt for the task to bridge the gap between the downstream task and the PLM's capability. PE shows promising results in various few-shot learning applications.

Schema of Comprehensive Instruction However, prompts in PE are often concise. To fully exploit the capability of PLMs, we propose to construct Comprehensive Instructions (CINS) on top of PE. The idea is to provide extra task-specific instructions for PLMs to understand critical *abilities* to solve the task. Besides the short prompts, we propose to add **task definition** and **task constraints** as instructions. An abstract configuration of CINS compared to PE and STD can be visualized in Figure 2. The goal of these two additional components are elaborated below:

- **Task Definition**: it provides a high-level natural language definition of the task itself. It describes the nature of the task, such as what types of input and output that the PLM is dealing with.

	Task Definition	Task Constraint	Prompt
IC	Predict the intent of the input query. Intent is the main topic or purpose of a query.	Model needs to select the most suitable intent from: {candidate labels + label descriptions}♣.	What is the intent of the given query?
DST	Predict the {slot description}♠ requested by User.	Select the most suitable value from: {candidate values}♢. If multiple values appear, select the latest one.	What is / Whether the {slot description}?
NLG♡	Verbalize the input representation / Paraphrase the input sentences.	The output should be natural and concise, and it should preserve the meaning and information of the input.	How to verbalize the input? / What is the paraphrase utterance of the input?

Table 1: ♣ this is the concatenation of all candidate intents with its corresponding descriptions. ♠ two types of descriptions for the slot "hotel-stars" can be founded in Table 5. ♢ candidate values for categorical slots (e.g. area, type), and it is left empty for open non-categorical slots (e.g. time, name). ♡ Two types of task definitions and prompts are used for Naive ($\mathcal{A}$) and T2G2 ($\mathcal{A}'$) representations mentioned in Section 3.1 respectively.

- **Task Constraint**: it gives more fine-grained task-specific constraint w.r.t output generated by the PLM. For example, the candidate labels and the descriptions of each candidate label. Task constraints aim to compliments the task definition to give more instructions about what the should the PLM output. It is not independent of the task definition, and it put more emphasis on the constraints w.r.t. the output space.

Different components of CINS are concatenated using a [SEP] token, and each component starts with a leading identifier (*"Input:", "Definition:", "Constraint:", "Prompt:"*).

CINS for ToD Tasks This section elaborates the realization of Comprehensive Instruction for three ToD tasks summarized in Table 1. We formulate the "prompt" using "Question" expressions, and the advantage over "Declarative" expressions will be analyzed in § 4.4. Next, we mainly explain "task definition" and "task constraint" for different tasks.

CINS for IC For intent classification, the definition explains the task followed by the meaning of "intent". To add constraints w.r.t. the output space, we include candidate intent labels and their corresponding descriptions in the constraint component. More specifically, for K candidate intents in a domain, we concatenate all intent names n_i with their descriptions d_i as $\{n_1 : d_1, \dots, n_1 : d_K\}$ to add to the constraint.

CINS for DST For dialog state tracking, the model predicts the value of different slots requested by the user from a dialog context. This task definition is encoded in our task definition component. "{slot description}" follows Lin et al. (2021) that encodes the type of a slot. In the task constraint component, we encode the constraint of candidate values for slots (e.g. *are, type, price-range*, etc.) that have categorical values (Zhang et al. 2020; Rastogi et al. 2020). For non-categorical slots (*name, time*) that have open values, no such constraint is enforced. Furthermore, a slot might be mentioned multiple times in a dialog history, and the correct value of interest often needs to be captured in its latest mention, and we also encode this information in the task constraint. Lastly, for slots (e.g. *has-internet, has-parking*) with "*yes/no*" values, we formulate the prompt question with "*Whether*", otherwise, the prompt question starts with "*What*".

CINS for NLG For natural language generation, the task definition is customized for two types of input representations (c.f. §3.1). For "**Naive**" representation, the task is to *verbalize* the semantic representation to a natural language utterance. For "**T2G2**" representation, *paraphrasing* is a more precise definition. Exact task definitions for these two cases are given in Table 1. In terms of the constraint, we tell the model the output utterance should be *natural and concise*, and it should also *preserve the meaning and information* of the original input representation for the fidelity concern.

4 Experiment

4.1 Few-shot Datasets

We evaluate three different ToD downstream tasks with three different datasets respectively.

OOS For intent classification, we use a benchmark dataset from Larson et al. (2019). Apart from the single out-of-scope intent, it contains 150 intents in 15 domains. Each domain contains 15 intents with 1,500/300/450 instances for train/validation/test, and data are balanced across different intents. Several domains are similar to each other, and we test on 5 representative domains (*Bank*, *Home*, *Travel*, *Utility*, and *Auto*). For few-shot setups, we sample k instances per intent from the training data, noted as "k-shot".

MultiWOZ2.0 We evaluate dialog state tracking task using MultiWOZ2.0 (Budzianowski et al. 2018). It contains 8,420/1,000/1,000 dialogues for train/validation/test spanning over 7 domains. Following Wu et al. (2019); Lin et al. (2021), we adopt *attraction*, *hotel*, *restaurant*, *train*, and *taxi* domains for training, as the test set only contains these 5 domains. In few-shot setups, we experiment with "k% Data", i.e. only k% of the training dialogs are used.

FewShotSGD Kale and Rastogi (2020a) is the version of the schema-guided-dataset (Rastogi et al. 2019) for natural language generation. The full train/validation/test sets contain 160k/24k/42k utterances. In "k-shot" experiments, k di-

T5-small		Average	Bank	Home	Travel	Utility	Auto
1-shot	STD	35.5 ± 1.9	32.2	29.0	25.0	47.6	43.6
	PE	41.7 ± 2.9	38.0	34.4	33.6	54.4	48.1
	CINS	**71.1** ± 4.4	**62.6**	**57.9**	**80.4**	**84.2**	**70.2**
5-shot	STD	72.4 ± 1.7	66.2	64.4	74.5	80.2	76.5
	PE	76.8 ± 2.5	71.9	65.8	83.0	84.1	79.5
	CINS	**85.6** ± 1.7	**82.4**	**71.3**	**94.5**	**93.3**	**86.2**
Full	STD	96.8	93.6	96.0	98.0	98.4	98.0

T5-base		Average	Bank	Home	Travel	Utility	Auto
1-shot	STD	56.0 ± 3.6	56.6	41.6	58.5	62.7	60.5
	PE	61.4 ± 3.1	62.3	48.7	59.2	74.9	62.0
	CINS	**79.2** ± 2.2	**80.8**	**60.2**	**87.3**	**86.2**	**81.5**
5-shot	STD	85.8 ± 2.1	83.3	72.1	91.1	93.8	89.0
	PE	87.0 ± 1.3	86.7	72.2	92.4	94.8	89.0
	CINS	**91.1** ± 2.2	**89.1**	**80.2**	**97.1**	**95.4**	**93.7**
Full	STd	97.4	94.7	96.7	98.1	98.7	98.5

Table 2: Accuracy in percentage [%] for intent classification task with T5-small (left) and T5-base (right). The "Average" column reports the average results and the standard deviations of 5 domains.

T5-small		Average	Attr.	Hotel	Rest.	Taxi	Train
1% Data	STD	33.6 ± 2.3	25.1	24.4	32.9	60.0	25.4
	PE	41.7 ± 3.6	36.0	25.9	**33.8**	59.9	52.3
	CINS	**43.1** ± 1.8	**42.0**	**27.3**	32.7	**60.4**	**53.1**
5% Data	STD	55.1 ± 2.2	54.3	43.0	50.2	59.0	69.1
	PE	55.7 ± 1.6	**57.0**	42.2	51.1	59.2	68.7
	CINS	**57.0** ± 1.1	56.9	**43.4**	**51.3**	**61.8**	**71.3**
Full	STD	72.0	71.3	59.5	68.0	81.4	80.0

T5-base		Average	Attr.	Hotel	Rest.	Taxi	Train
1% Data	STD	45.5 ± 3.5	41.5	30.8	36.7	58.4	60.1
	PE	46.5 ± 3.1	41.7	31.3	39.0	59.5	**61.2**
	CINS	**47.9** ± 2.1	**45.6**	**33.9**	**40.6**	**59.7**	60.3
5% Data	STD	58.8 ± 1.7	59.7	44.5	**54.1**	62.8	73.2
	PE	57.8 ± 2.9	59.3	43.7	51.9	61.7	72.6
	CINS	**59.7** ± 2.4	**61.2**	**46.2**	53.9	**63.3**	**73.8**
Full	STD	72.8	73.6	60.5	66.4	81.9	81.8

Table 3: Joint Goal Accuracy in percentage [%] for few-shot dialog state tracking using T5-small (left) and T5-base (right). The "Average" column reports the average results and the standard deviations of 5 domains.

alogs from each 14 training domains are sampled from the training data. We use the same 5/10-shot training split as in Kale and Rastogi (2020a) because they both contain utterances for every dialog act and slot present in the full training set.[1]

To test "realistic few-shot learning" scenarios mentioned before, we down-sample validation data to be the same size as the few-shot training data in all our experiments.

4.2 Experiment Settings

We tested **T5-small** (60M parameters, 6 encoder-decoder layers) as well as **T5-base** (110M parameters, 12 encoder-decoder layers) using the huggingface repository.[2] All models are trained using AdamW (Loshchilov and Hutter, 2018) optimizer with the initial learning rate of 1e-4 for DST and NLG, and 3e-4 for IC. In all experiments, we train the models with batch size 8 for 30 epochs for IC, 20 epochs for DST, and 50 epochs for NLG. Early stop according to the loss on the validation set. In the testing phase, we use greedy decoding. We use 4 NVIDIA V100 GPUs for all of our experiments.

For comparison, we consider two baselines, STD and PE. For prompting based method, both PE and CINS, six prompts are tested, and we report the results with the best prompt choice without mentioned specifically. For STD, we also report a *upper bound* using all labeled training data and validation data, referred to as "Full". For all few-shot experiments, we report mean and standard deviation with three different random seeds to reduce training/validation data sampling variance.

[1] 1-shot training data is not provided with such property

[2] https://huggingface.co

4.3 Main Experiment Results

Intent Classification Accuracy of few-shot intent classification on 5 domains over OOS is presented in Table 2. For both T5-small and T5-base, 1-shot and 5-shot settings are considered, and the column headed with "Average" averages results of 5 domains. We see that STD performs the worst in different configurations and that PE consistently outperforms STD. CINS significantly outperforms both STD and PE in all configurations, especially with fewer label (1-shot). In 1-shot setting, CINS achieves 29.4% and 17.8% higher average accuracy than PE for T5-small and T5-base respectively. In 5-shot settings, the above two margins are 8.8% and 4.1%. These results demonstrate that CINS effectively boost the few-shot learning capability for intent classification.

Dialog State Tracking Results of DST on MultiWOZ2.0 are presented in Table 3. 1% and 5% labeled data are tested w.r.t. T5-small and T5-base. The common evaluation metrics joint goal accuracy (JGA) (Budzianowski et al. 2018; Wu et al. 2019) is used, which checks whether the predicted states (domain, slot, value) match the ground truth states given a context. PE on average performs better than STD except for the configuration with 5% labeled data using T5-base. We see that CINS consistently improves the averaged JGA over both STD and PE in different configurations. For example, CINS has 3.1% and 9.5% average JGA improvement over PE and STD respectively when 1% labeled data

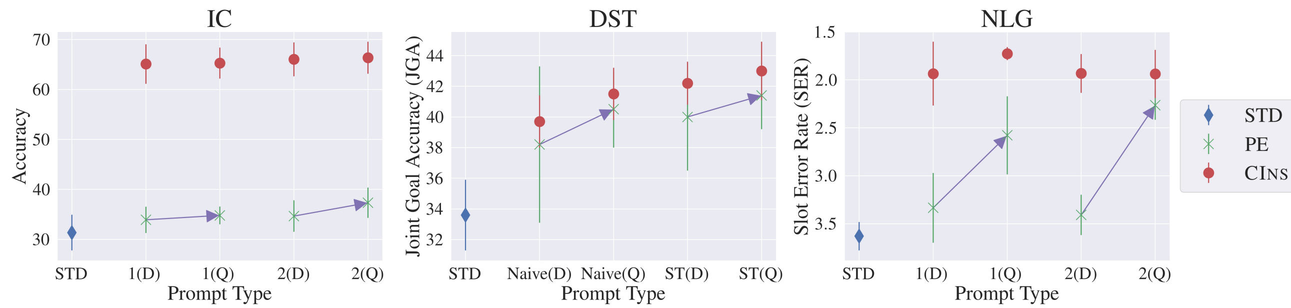

Figure 3: Results of IC (1-shot), DST (1% data), NLG (1-shot) with different types of prompts with T5-small. Different prompts tested for IC and DST are provided in Table 5. For the same prompt backbone, the version with **(D)** stands for a **Declarative** expression, while the version with **(Q)** stands for a **Question** expression. Standard deviations over different random seeds are also plotted.

T5-small			**SER** ↓	**BLEU** ↑
Naive	5-shot	STD	10.2 ± 0.49	17.3 ± 0.23
		PE	9.8 ± 0.43	17.1 ± 0.17
		CINS	$\mathbf{6.9} \pm 0.25$	$\mathbf{17.5} \pm 0.13$
	10-shot	STD	5.9 ± 0.30	19.0 ± 0.18
		PE	5.2 ± 0.34	19.2 ± 0.26
		CINS	$\mathbf{4.6} \pm 0.28$	$\mathbf{19.4} \pm 0.07$
	Full	STD	1.0	26.3
T2G2	5-shot	STD	3.6 ± 0.15	25.5 ± 0.05
		PE	2.3 ± 0.15	26.0 ± 0.05
		CINS	$\mathbf{1.7} \pm 0.07$	$\mathbf{26.3} \pm 0.11$
	10-shot	STD	3.5 ± 0.15	25.9 ± 0.07
		PE	1.9 ± 0.33	26.4 ± 0.12
		CINS	$\mathbf{1.3} \pm 0.07$	$\mathbf{26.7} \pm 0.04$
	Full	STD	0.4	28.6

Table 4: Performance of few-shot natural language generation using T5-small as the generation model. Two types of semantic representations are tested: Naive and T2G2.

are used with T5-small. Smaller margins can be observed for other configurations.

Natural Language Generation Results of T5-small for 5-shot and 10-shot NLG using two types of semantic representations "Naive" and "T2G2" are included in Table 4. We only report T5-small as it already performs well enough. Following prior works (Wen et al. 2015; Kale and Rastogi 2020a), we use BLEU (Papineni et al. 2002) and Slot Error Rate (SER Dušek and Jurcicek (2019)) as metrics. SER measures the fraction of generated texts where at least one slot was not correctly copied from the structured data. CINS outperforms both PE and STD in all configurations and metrics with notable margins. Compared to PE, CINS improves SER of "Naive" by 2.9% and 0.6% in 5-shot and 10-shot settings respectively. These two margins are 0.6% and 0.6% for "T2G2". The improvement margins over STD are larger. Moreover, when "T2G2" is used as the input representation, CINS with only 5-shot or 10-shot training samples achieves comparable performance compared to "Full".

Altogether, our experiments on three different downstream tasks reveal that:

- Comprehensive Instruction provides complimentary benefits over standard input and prompting. CINS consistently improves both PE and STD in all configurations of three few-shot ToD tasks. The margin is evident on IC, indicated by 17-30% average gain over PE with 1-shot data; 4-9% gain with 5-shot data. The margin is smaller on two more challenging DST and NLG tasks.
- Comprehensive Instruction bridges the gap between few-shot learning and full supervision. STD and PE with few-shot labeled data perform much worse than models trained with all labeled data ("Full") for IC and NLG. CINS largely improves performances on these two tasks with results comparable to "Full".

4.4 Analysis and Discussions

What is a good prompt? In this experiment, we study what makes it a good prompt with T5 used as the backbone model. In Table 5, we present two types of "prompt roots" highlighted in different colors for IC, DST and NLG ("T2G2"). For each prompt root (row), two expressions are considered in declarative and question forms.[3] In Figure 3, we present results of using these different prompts with T5-small for IC (1-shot), DST (1% Labeled Data) and NLG (5-shot). Results for PE are illustrated by green points. For the same prompt root, prompts with a question (Q) expressions always outperform prompt with declarative (D) expressions for both PE and CINS. Such a pattern is more obvious for PE, and it can be better visualized by purple arrows.

Does CINS improve different prompts? In this experiment, we study whether CINS achieves consistent improvements with different prompts. In Figure 3, we compare CINS and PE with four prompts mentioned in Table 5. For each prompt, regardless of declarative or questions expressions, CINS outperforms PE. This results validates that adding task-specific definition and constraint as instructions is beneficial for prompts in PE.

[3]A prefix "Question:" is adopted as we found that adding it achieves slightly better performance because this prefix is commonly seen during the T5 pre-training phase.

	Prompt	Declarative (D)	Question (Q)
IC	1	The given query asks about:	Question: What does the given query ask about?
	2	The intent of the given query is:	Question: What is the intent of the given query?
DST (hotel_stars)	Naive†	stars of the hotel	Question: What is the stars of the hotel?
	Slot Type (ST)‡	number of stars of the hotel	Question: What is the number of stars of the hotel?
NLG	1	Paraphrase the input:	Question: what is the paraphrase of the input?
	2	Rewrite the input:	Question: what is the rewriting of the input?

Table 5: Different prompts for IC, DST and NLG ("T2G2"). In each row, two expressions (declarative and question) are considered for a highlighted prompt root. For DST, we use the slot "hotel-stars" as an example, and more descriptions can be founded in Lin et al. (2021). † Transform "domain-slot" to "[slot] of the [domain]. ‡ Transform "domain-slot" to "[slot type] [slot] of the [domain]" (Lin et al. 2021).

T5-small	**IC**	**DST**	**NLG**
CINS	71.1	43.1	1.7
w/o Description	65.6	41.2	-
w/o Definition	69.8	42.5	1.9
w/o Prompt	70.1	42.2	2.3
w/o Constraint	45.7	40.8	2.4
PE	41.7	40.0	2.6

Table 6: Ablation Study for CINS for IC (1-shot), DST (1% data), NLG (5-shot SER) with T5-small. "Description" stands for IC label and DST slot descriptions.

Is CINS robust? For main experiments conducted in § 4.3, we experiment with different data sizes and model sizes for IC and DST; different data sizes and input forms for NLG. In total, we have 12 configurations (4 for IC, 4 for DST, 4 SERs for NLG). For each configuration, standard deviations of three random seeds are also reported. We could see that CINS achieves the lowest standard deviation in 9/12 configurations. This result demonstrates that CINS is more robust to the data sampling variance in few-shot learning settings. Furthermore, we could see from Figure 3 that the performance of CINS is also less sensitive to the prompts than PE. For example, the performance of 2(D) vs. 2(Q) or ST(D) vs. ST(Q) differs a lot for PE, while they perform similarly for CINS. Therefore, We contend that extra task-specific instructions in CINS improve model robustness w.r.t. the choice of few-shot training data as well as prompt.

Ablation study In Table 6, we compare several simplified versions of CINS to understand the effects of different components. "w/o Definition", "w/o Constraint", and "w/o Prompt" are intuitive. "w/o Description" for IC removes label descriptions in constraint, and "w/o Description" for DST replace the slot description from "Slot Type" to "Naive" (Lin et al. 2021). We observe that: (i) label and slot descriptions are beneficial. Removing it degrades performance by 5.5% and 1.9% on IC and DST respectively. (ii) Task definition and Prompt are both concise but advantageous. Dropping either of them ("w/o Definition", "w/o Prompt") hurts performance slightly. (iii) Task-specific constraint is a critical component, indicated by relatively large performance drop on three tasks when removing it ("w/o Constraint"). Nevertheless, it still outperforms "PE", meaning that the model still learns from task definitions.

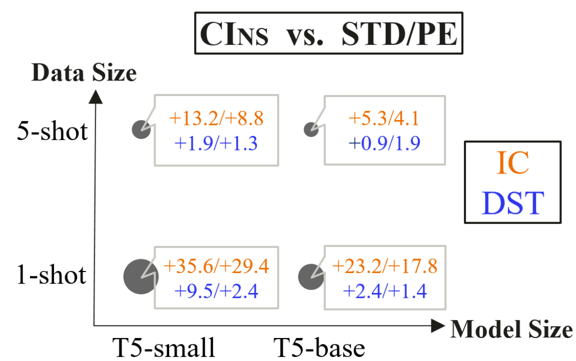

Figure 4: Accuracy improvements of CINS over STD/PE in different configurations of intent classification (orange) and dialog state tracking (blue) tasks.

Effect of Different Model and Data Sizes In Figure 4, we plot the improvements of CINS over STD and PE in different configurations of IC (red) and DST (blue). We could see that the improvement margin of CINS over STD/PE is largest in the 1-shot setting with T5-small for both IC and DST. When more labeled data are used or the model size is increased, improvement margins get smaller. The only exception is the 5-shot margin of CINS over PE for T5-base on DST, because this is the only case when PE underperforms STD. Therefore, we contend that CINS is especially beneficial for low-resource learning with reasonable-sized models.

5 Conclusion

We study how to instruct PLMs for few-shot learning in ToD. CINS is proposed to augment state-of-the-art prompting techniques with extra task-specific definition and constraint. Extensive empirical results on three ToD tasks demonstrate the consistent improvements of CINS. Our findings on using instructions may inspire future studies towards better utilizing PLMs for building more sample-efficient and scalable ToD systems.

References

Brown, T. B.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D. M.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. *CoRR*, abs/2005.14165.

Budzianowski, P.; and Vulic, I. 2019. Hello, It's GPT-2 - How Can I Help You? Towards the Use of Pretrained Language Models for Task-Oriented Dialogue Systems. In *NGT@EMNLP-IJCNLP*, 15–22. Association for Computational Linguistics.

Budzianowski, P.; Wen, T.; Tseng, B.; Casanueva, I.; Ultes, S.; Ramadan, O.; and Gasic, M. 2018. MultiWOZ - A Large-Scale Multi-Domain Wizard-of-Oz Dataset for Task-Oriented Dialogue Modelling. In *EMNLP*, 5016–5026. Association for Computational Linguistics.

Cai, F.; Zhou, W.; Mi, F.; and Faltings, B. 2021. SLIM: Explicit Slot-Intent Mapping with BERT for Joint Multi-Intent Detection and Slot Filling. *arXiv preprint arXiv:2108.11711*.

Chao, G.-L.; and Lane, I. 2019. BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer. *Proc. Interspeech 2019*, 1468–1472.

Chen, Q.; Zhuo, Z.; and Wang, W. 2019. Bert for joint intent classification and slot filling. *arXiv preprint arXiv:1902.10909*.

Devlin, J.; Chang, M.; Lee, K.; and Toutanova, K. 2019. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In *NAACL-HLT (1)*, 4171–4186. Association for Computational Linguistics.

Dong, L.; Yang, N.; Wang, W.; Wei, F.; Liu, X.; Wang, Y.; Gao, J.; Zhou, M.; and Hon, H. 2019. Unified Language Model Pre-training for Natural Language Understanding and Generation. In *NeurIPS*, 13042–13054.

Dušek, O.; and Jurcicek, F. 2019. Neural Generation for Czech: Data and Baselines. In *Proceedings of the 12th International Conference on Natural Language Generation*, 563–574.

Gao, T.; Fisch, A.; and Chen, D. 2021. Making Pre-trained Language Models Better Few-shot Learners. In *ACL/IJCNLP (1)*, 3816–3830. Association for Computational Linguistics.

Ham, D.; Lee, J.; Jang, Y.; and Kim, K. 2020. End-to-End Neural Pipeline for Goal-Oriented Dialogue Systems using GPT-2. In *ACL*, 583–592. Association for Computational Linguistics.

Hosseini-Asl, E.; McCann, B.; Wu, C.; Yavuz, S.; and Socher, R. 2020. A Simple Language Model for Task-Oriented Dialogue. *CoRR*, abs/2005.00796.

Hu, S.; Ding, N.; Wang, H.; Liu, Z.; Li, J.; and Sun, M. 2021. Knowledgeable Prompt-tuning: Incorporating Knowledge into Prompt Verbalizer for Text Classification. *CoRR*, abs/2108.02035.

Jiang, Z.; Xu, F. F.; Araki, J.; and Neubig, G. 2020. How Can We Know What Language Models Know. *Trans. Assoc. Comput. Linguistics*, 8: 423–438.

Kale, M.; and Rastogi, A. 2020a. Template Guided Text Generation for Task Oriented Dialogue. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 6505–6520. Online: Association for Computational Linguistics.

Kale, M.; and Rastogi, A. 2020b. Text-to-Text Pre-Training for Data-to-Text Tasks. In *Proceedings of the 13th International Conference on Natural Language Generation*, 97–102.

Larson, S.; Mahendran, A.; Peper, J. J.; Clarke, C.; Lee, A.; Hill, P.; Kummerfeld, J. K.; Leach, K.; Laurenzano, M. A.; Tang, L.; and Mars, J. 2019. An Evaluation Dataset for Intent Classification and Out-of-Scope Prediction. In *EMNLP/IJCNLP (1)*, 1311–1316. Association for Computational Linguistics.

Lester, B.; Al-Rfou, R.; and Constant, N. 2021. The Power of Scale for Parameter-Efficient Prompt Tuning. *CoRR*, abs/2104.08691.

Lewis, M.; Liu, Y.; Goyal, N.; Ghazvininejad, M.; Mohamed, A.; Levy, O.; Stoyanov, V.; and Zettlemoyer, L. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Li, X. L.; and Liang, P. 2021. Prefix-Tuning: Optimizing Continuous Prompts for Generation. In *ACL/IJCNLP (1)*, 4582–4597. Association for Computational Linguistics.

Lin, Z.; Liu, B.; Moon, S.; Crook, P. A.; Zhou, Z.; Wang, Z.; Yu, Z.; Madotto, A.; Cho, E.; and Subba, R. 2021. Leveraging Slot Descriptions for Zero-Shot Cross-Domain Dialogue StateTracking. In *NAACL-HLT (1)*, 5640–5648. Association for Computational Linguistics.

Liu, J.; Shen, D.; Zhang, Y.; Dolan, B.; Carin, L.; and Chen, W. 2021a. What Makes Good In-Context Examples for GPT-3? *CoRR*, abs/2101.06804.

Liu, P.; Yuan, W.; Fu, J.; Jiang, Z.; Hayashi, H.; and Neubig, G. 2021b. Pre-train, Prompt, and Predict: A Systematic Survey of Prompting Methods in Natural Language Processing. *CoRR*, abs/2107.13586.

Liu, X.; Zheng, Y.; Du, Z.; Ding, M.; Qian, Y.; Yang, Z.; and Tang, J. 2021c. GPT Understands, Too. *CoRR*, abs/2103.10385.

Mi, F.; Chen, L.; Zhao, M.; Huang, M.; and Faltings, B. 2020. Continual Learning for Natural Language Generation in Task-oriented Dialog Systems. In *EMNLP (Findings)*, volume EMNLP 2020 of *Findings of ACL*, 3461–3474. Association for Computational Linguistics.

Mi, F.; Huang, M.; Zhang, J.; and Faltings, B. 2019. Meta-Learning for Low-resource Natural Language Generation in Task-oriented Dialogue Systems. In *IJCAI*, 3151–3157. ijcai.org.

Mi, F.; Zhou, W.; Kong, L.; Cai, F.; Huang, M.; and Faltings, B. 2021. Self-training Improves Pre-training for Few-shot Learning in Task-oriented Dialog Systems. In *EMNLP (1)*, 1887–1898. Association for Computational Linguistics.

Mishra, S.; Khashabi, D.; Baral, C.; and Hajishirzi, H. 2021. Natural Instructions: Benchmarking Generalization to New Tasks from Natural Language Instructions. *CoRR*, abs/2104.08773.

Papineni, K.; Roukos, S.; Ward, T.; and Zhu, W.-J. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 311–318.

Peng, B.; Li, C.; Li, J.; Shayandeh, S.; Liden, L.; and Gao, J. 2020a. SOLOIST: Few-shot Task-Oriented Dialog with A Single Pre-trained Auto-regressive Model. *CoRR*, abs/2005.05298.

Peng, B.; Zhu, C.; Li, C.; Li, X.; Li, J.; Zeng, M.; and Gao, J. 2020b. Few-shot Natural Language Generation for Task-Oriented Dialog. In *EMNLP (Findings)*, 172–182. Association for Computational Linguistics.

Qin, G.; and Eisner, J. 2021. Learning How to Ask: Querying LMs with Mixtures of Soft Prompts. In *NAACL-HLT*, 5203–5212. Association for Computational Linguistics.

Radford, A.; Narasimhan, K.; Salimans, T.; and Sutskever, I. 2018. Improving language understanding by generative pre-training.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8): 9.

Raffel, C.; Shazeer, N.; Roberts, A.; Lee, K.; Narang, S.; Matena, M.; Zhou, Y.; Li, W.; and Liu, P. J. 2020. Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. *Journal of Machine Learning Research*, 21: 1–67.

Rastogi, A.; Zang, X.; Sunkara, S.; Gupta, R.; and Khaitan, P. 2019. Towards Scalable Multi-domain Conversational Agents: The Schema-Guided Dialogue Dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Rastogi, A.; Zang, X.; Sunkara, S.; Gupta, R.; and Khaitan, P. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, 8689–8696.

Schick, T.; and Schütze, H. 2020. Few-Shot Text Generation with Pattern-Exploiting Training. *CoRR*, abs/2012.11926.

Schick, T.; and Schütze, H. 2021a. Exploiting Cloze-Questions for Few-Shot Text Classification and Natural Language Inference. In *EACL*, 255–269. Association for Computational Linguistics.

Schick, T.; and Schütze, H. 2021b. It's Not Just Size That Matters: Small Language Models Are Also Few-Shot Learners. In *NAACL-HLT*, 2339–2352. Association for Computational Linguistics.

Tam, D.; Menon, R. R.; Bansal, M.; Srivastava, S.; and Raffel, C. 2021. Improving and Simplifying Pattern Exploiting Training. *CoRR*, abs/2103.11955.

Wen, T.-H.; Gasic, M.; Mrkšić, N.; Su, P.-H.; Vandyke, D.; and Young, S. 2015. Semantically Conditioned LSTM-based Natural Language Generation for Spoken Dialogue Systems. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, 1711–1721.

Williams, R. J.; and Zipser, D. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, 1(2): 270–280.

Wu, C.; Madotto, A.; Hosseini-Asl, E.; Xiong, C.; Socher, R.; and Fung, P. 2019. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. In *ACL (1)*, 808–819. Association for Computational Linguistics.

Wu, C.-S.; Hoi, S.; Socher, R.; and Xiong, C. 2020. TOD-BERT: Pre-trained natural language understanding for task-oriented dialogues. In *EMNLP*, 917–929. Association for Computational Linguistics.

Zhang, J.; Hashimoto, K.; Wu, C.-S.; Wang, Y.; Philip, S. Y.; Socher, R.; and Xiong, C. 2020. Find or Classify? Dual Strategy for Slot-Value Predictions on Multi-Domain Dialog State Tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, 154–167.