Dialogue Summaries as Dialogue States (DS2), Template-Guided Summarization for Few-shot Dialogue State Tracking

Jamin Shin[1]*, **Hangyeol Yu**[1]*, **Hyeongdon Moon**[1]*, **Andrea Madotto**[2], **Juneyoung Park**[1]
[1]Riiid AI Research
[2]The Hong Kong University of Science and Technology
jayshin.nlp@gmail.com, {hangyeol.yu,hyeongdon.moon}@riiid.co
amadotto@connect.ust.hk, juneyoung.park@riiid.co

Abstract

Annotating task-oriented dialogues is notorious for the expensive and difficult data collection process. Few-shot dialogue state tracking (DST) is a realistic solution to this problem. In this paper, we hypothesize that dialogue summaries are essentially unstructured dialogue states; hence, we propose to reformulate dialogue state tracking as a dialogue summarization problem. To elaborate, we train a text-to-text language model with synthetic template-based dialogue summaries, generated by a set of rules. Then, the dialogue states can be recovered by inversely applying the summary generation rules. We empirically show that our method DS2 outperforms previous works on few-shot DST in MultiWoZ 2.0 and 2.1, in both cross-domain and multi-domain settings. Our method[1] also exhibits vast speedup during both training and inference as it can generate all states at once. Finally, based on our analysis, we discover that the naturalness of the summary templates plays a key role for successful training.

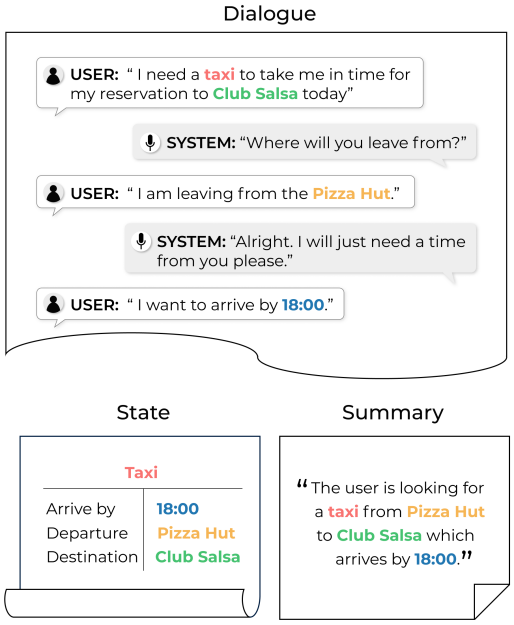

Figure 1: Example dialogue in taxi domain, its dialogue state, and template summary created from the state.

1 Introduction

Task-oriented dialogue systems (TOD) have penetrated our daily lives much more than before and they will continue to increase their presence. For example, many of our mobile devices are equipped with such dialogue agents like Siri, and we now often encounter customer service or flight reservation bots. Dialogue State Tracking (DST) is an essential element of such task-oriented dialogue systems (Wu et al., 2019; Balaraman et al., 2021). The main goal of this component is to understand the user's requirements expressed during the conversation under a given schema or ontology. Hence, as shown in Figure 1, accurately extracting the departure, destination, and arrival time of the user is key to creating a good user experience.

However, collecting such turn-level dialogue state annotations is very expensive and requires a significant amount of design and mediating efforts from domain experts (Budzianowski et al., 2018; Eric et al., 2020; Park et al., 2021). This is because the collection process follows the Wizard-of-Oz (WoZ) style (Kelley, 1984), which requires two human workers to converse with each other and annotate the states for each turn. To cope with this inherent scalability issue, Budzianowski et al. (2018) attempted to crowd-source this process in MultiWoZ 2.0 which resulted in one of the largest publicly available multi-domain task-oriented dialogue dataset. However, the resulting dataset is very noisy in data annotations which often hindered the training and evaluation process. In fact, the

* Equal Contribution: JS proposed the main idea and scaled up the experiments. HY designed and implemented the heuristic state tracking component. HM conducted rapid prototyping, analysis, and ablations.

[1]Code: github.com/jshin49/ds2

Findings of the Association for Computational Linguistics: ACL 2022, pages 3824 - 3846

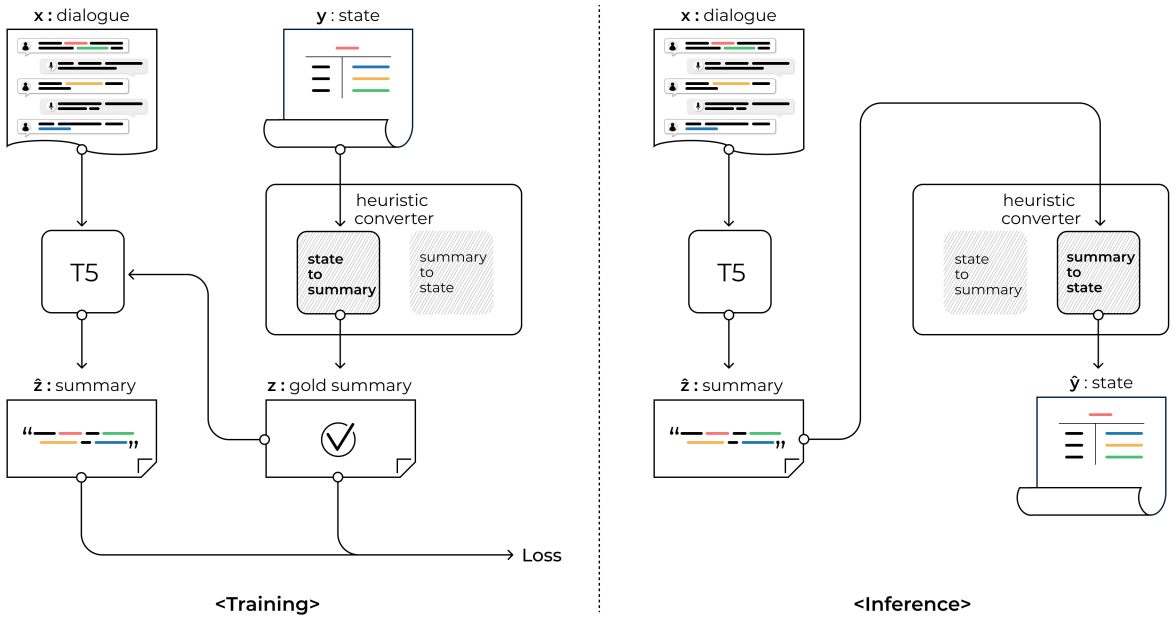

Figure 2: Overall picture of our method DS2.

community has already seen 4 different revisions of this dataset from 2.1 to 2.4 (Eric et al., 2020; Zang et al., 2020; Han et al., 2021; Ye et al., 2021).

Furthermore, in realistic industrial settings, having to expand the existing model and ontology to include new domains and slot-values is a common phenomenon. Naturally, there have been many recent works that proposed (zero) few-shot settings to rely on less annotated data. For instance, both STARC (Gao et al., 2020) and TransferQA (Lin et al., 2021a) achieve great few-shot DST performance on MultiWoZ 2.0 by prompting large pre-trained language models like BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020) with natural language questions (e.g. "In what area is the user looking for a hotel?").

Meanwhile, despite the good performance of the aforementioned works, they still suffer from certain issues. **1) They often require** a large amount of expensive labeled training data from other tasks or domains for task-specific pre-training. For example, as shown in Table 1, SOLOIST (Peng et al., 2021) uses $\sim 766K$, TOD-BERT (Wu et al., 2020a) uses $\sim 1.39M$, and PPTOD (Su et al., 2021) utilizes $\sim 2.37M$ dialogue utterances. Meanwhile, TransferQA (Lin et al., 2021a) also uses a vast amount of QA data ($\sim$720K). **2) QA-style prompting** as in TransferQA (Lin et al., 2021a) not only requires additional efforts to handle "none" and "yes-no" slots but also has an expensive slot-value decoding time complexity: k times inference of a language model where k is the number of slots. Overall, the aforementioned works are still expensive in terms of time, money, and engineering costs.

Addressing the above challenge, we propose to cast Dialogue State Tracking as a Dialogue Summarization task; hence the name is **Dialogue Summaries as Dialogue States (DS2)**. The main hypothesis for this reformulation is that *dialogue summaries are essentially unstructured dialogue states*. In this paper, we explore this reformulation to the limit. We fine-tune large text-to-text pre-trained language models (e.g. T5, BART) with synthetic dialogue summaries. These summaries are created by heuristic rules from the dialogue states, as in Figure 1. Hence, as these models already excel in text summarization, the research question we ask is *whether we can guide dialogue summarization models to generate dialogue summaries that conform to the templates we provide*. Then, we can extract the dialogue states by inversely applying the rules we used to create the synthetic summaries.

Compared to previous approaches, our method has several advantages that come naturally. First, we easily reduce the pre-train & fine-tune discrepancies without any DST-specific engineering by leveraging dialogue summarization datasets. These datasets are an order of magnitude smaller in annotated data size (e.g. SAMSum (Gliwa et al., 2019) has $\sim 200K$ utterances). Second, we achieve great speedup in both training and inference because we only need to summarize once, and we can extract

Model	# of Pre-train Data	Data Type
TOD-BERT(Wu et al., 2020a)	∼1.39M	Dialogue Utterances
PPTOD (Su et al., 2021)	∼2.37M	Dialogue Utterances
Transfer QA (Lin et al., 2021a)	∼720K	QA Pairs
SOLOIST (Peng et al., 2021)	∼766K	Dialogue Utterances
Ours - DS2	∼199K	Dialogue Utterances

Table 1: Pre-train data usage scale comparison with other models. We used SAMSum (Gliwa et al., 2019), which is a dialogue summarization dataset, and we estimated the number of utterances in SAMSum to be in the range (154k, 243k).

slot values from the summary with negligible cost.

Finally, the significant improvement that DS2 brings to MultiWoZ 2.0 and 2.1 datasets in the few-shot DST performance for both *cross-domain* and *multi-domain* settings empirically show the effectiveness of our approach. *Without extensively using such expensive annotated data for pre-training*, DS2 generally outperforms previous works that do so. In our analysis, we also show how naturalness of the summary has played a key role for this work. Our main contribution can be summarized as such:

- We propose DS2, which is the first approach to cast Dialogue State Tracking as Dialogue Summarization.
- Our formulation provides relatively easier reduction of pre-train & fine-tune discrepancy, while also significantly improving training and inference speed for generative DST.
- We empirically show that our method outperforms previous methods in MultiWoZ 2.0 and 2.1 for both cross-domain and multi-domain few-shot DST settings.

2 Related Work

Dialogue State Tracking is a well-known subtask of task oriented dialog systems (Williams and Young, 2007; Williams et al., 2014). The current state-of-the-art techniques fine-tune pre-trained language models (Lei et al., 2018; Zhang et al., 2020c; Wu et al., 2020a; Peng et al., 2021; Zhang et al., 2020a; Kim et al., 2020a; Lin et al., 2020; Chen et al., 2020; Heck et al., 2020; Mehri et al., 2020; Hosseini-Asl et al., 2020; Yu et al., 2021; Li et al., 2021a) are often further trained with a large amount of annotated data.

Few-Shot DST is a promising direction for reducing the need of human annotation while achieving quasi-SOTA performance with a fraction of the training data. Different techniques have been proposed (Wu et al., 2019; Mi et al., 2021; Li et al., 2021b; Gao et al., 2020; Lin et al., 2021b,a; Campagna et al., 2020; Wu et al., 2020b; Su et al., 2021; Peng et al., 2021; Wu et al., 2020a). We briefly describe and compare DS2 with existing few-shot models in Section 4.5.

Dialogue Summarization The community has been seeing an increasing amount of interest in this subfield: from datasets (Zhu et al., 2021; Zhong et al., 2021; Chen et al., 2021; Fabbri et al., 2021; Zhang et al., 2021) to models (Wu et al., 2021; Feng et al., 2021; Khalifa et al., 2021; Chen and Yang, 2020).

Prompt Engineering Many recent works on prompt engineering or Pattern Exploiting Training (PET) (Schick and Schütze, 2021b,a,c; Gao et al., 2021; Liu et al., 2021; Madotto et al., 2021; Shin et al., 2021; Liu et al., 2021) have been proposed to explore prompt-based few-shot learning capabilities for Pre-trained Language Models. Interestingly, they share similar insights about the critical role of natural templates for successful few-shot learning.

3 Methodology

3.1 Background

A data point for DST is a pair of a task-oriented dialogue $\mathbf{x}$ and a sequence $\{\mathbf{y}_t\}_{t=1}^{n}$ of dialogue states where t and n refer to current turn index and the total number of turns in the dialogue respectively. Here, $\mathbf{y}_t$ denotes the dialogue state after turn t. A dialogue state is a set of slot-value pairs,

$$\mathbf{y}_t = \{(k_1, v_1), (k_2, v_2), \ldots, (k_m, v_m)\}$$

where the set of all possible slots k_i's in a domain is predefined. For example, the attraction domain in MultiWoZ has three kinds of slots, namely, 'attraction-area', 'attraction-name', and 'attraction-type'. With this setting, DST is a task to predict $\mathbf{y}_t$ given the truncated dialogue $\mathbf{x}_{1:t}$ as input for every t. For convenience, we will omit the turn index t.

3.2 Overview: Dialogue Summaries as Dialogue States (DS2)

In this section, we describe the overall picture of the proposed method, DS2. First, our method is composed of 3 components: the Pre-trained text-to-text Language Model (PLM; θ) such as T5, dialogue summary generator (*state-to-summary*; ϕ), and dialogue state extractor (*summary-to-state*; η). To briefly describe the training process, given a

dialogue $\mathbf{x}$, we first generate a synthetic summary $\mathbf{z} = \phi(\mathbf{y})$ as in Table 2, using the *state-to-summary* module. Instead of generating dialogue states directly as done by Wu et al. (2019); Gao et al. (2019), we fine-tune the PLM to predict $\mathbf{z}$. The training loss is then calculated between $\mathbf{z}$ and $\hat{\mathbf{z}}$, which is the cross-entropy loss between $P(\mathbf{z} \mid \mathbf{x})$ and the predicted summary $\mathbf{z}$. This process is described in the <Training> part of Figure 2 (left section). Note that the only module that we train is the summary model θ. During inference, the PLM generates a summary $\hat{\mathbf{z}}$ and the dialogue state $\hat{\mathbf{y}}$ is extracted from it using the *summary-to-state* module η. The right section of Figure 2 describes this process.

Our method DS2 reformulates DST into a summarization task. The idea is simple. If a model summarizes a given dialogue with all the slot-value information, *exactly in the format we want*, then we can simply use regular expressions to parse the slot-values from the generated summary. The mathematical assumption here is that the *state-to-summary* converter ϕ is a left inverse of the *summary-to-state* converter η. That is, $\eta(\phi(\mathbf{y}')) = \mathbf{y}'$ for every dialogue state $\mathbf{y}'$. Let $(\mathbf{x}, \mathbf{y})$ be a training sample. If a predicted summary $\hat{\mathbf{z}} = \theta(\mathbf{x})$ exactly matches the generated one $\mathbf{z} = \phi(\mathbf{y})$, the later step is straight forward by η as follows:

$$\eta(\theta(\mathbf{x})) = \eta(\hat{\mathbf{z}}) = \eta(\mathbf{z}) = \eta(\phi(\mathbf{y})) = \mathbf{y}.$$

Here, $\eta \circ \theta$ is the DST model we want.

Note that the space of all texts is larger than the set of all dialogue states defined by the ontology. The former is infinite but the latter is finite. Therefore, there is no one-to-one correspondence between two sets. That is one reason we consider a certain template for summaries: to restrict the set of candidate summaries so that the size perfectly matches the set of all states. One more benefit from the template is that it naturally provides a structural summary-to-state conversion.

Meanwhile, the reduced summarization task is subtle because, at least, a generated summary $\theta(\mathbf{x})$ must satisfy the template to guarantee our argument. In mathematical words, $\theta(\mathbf{x})$ should be in the image of ϕ. In general, it is nontrivial to control a deep learning model so that its output is always in an arbitrary subset, and it is even harder with few samples. Therefore, we hypothesize that the naturalness of the template is a key factor to the performance of our model.

Slot Name	**Slot Template**	**Slot Value**
attraction-area	located in the _	*center*
attraction-name	called _	*byard art*
attraction-type	which is a _	*museum*
Sentence Prefix	The user is looking for an attraction	
Example Synthetic Summary	"The user is looking for an attraction called *byard art* which is a *museum* located in the *center*."	

Table 2: Template for attraction domain in MultiWoZ.

3.3 State-to-summary Converter

For each dialogue domain, we manually wrote a template to automatically synthesize human-readable summaries from dialogue states. Designing the template, domain-specific information is considered such as the name of slots and possible values. Table 2 illustrates a template for the "attraction" domain in MultiWoZ with example values. This template itself can be regarded as the previously discussed function ϕ, which takes a dialogue state as an input to produce a dialogue summary.

Given a state, the corresponding summary is built based on the template in a hierarchical manner. Suppose there are m slots in the current domain, namely, $k_1, \ldots, k_m$. We define a phrase template p_i for each slot k_i, which is a function that takes a value string as input and produces a phrase. In Table 2, the slot named "attraction-area" is mapped to a phrase template "located in the _". After combining with the slot-value *centre*, we get a phrase "located in the *centre*". Let $\mathbf{y} = \{(k_1, v_1), \ldots, (k_{m'}, v_{m'})\}$ be a given state where $m' \leq m$. Each value v_i of a slot appearing in the state is matched to the phrase template p_i, so we get the set of phrases $\{p_1(v_1), \ldots, p_{m'}(v_{m'})\}$. They are joined together and added to the sentence prefix of the domain such as "The user is looking for an attraction", to get the final summary:

> "The user is looking for an attraction called *byard art* which is a *museum* located in the *centre*."

The template also covers exceptional cases, *dontcare*. Each slot has a special phrase for *dontcare*. For example, "attraction-area" is mapped to a phrase "the location". For that case, another sentence prefix ", and he does not care about" is used. The resulting summary is:

> "The user is looking for an attraction which is a *museum*, and he does not care about *the location*."

We do not care too much about *none* values as it is naturally covered. Since we remove all slots whose values are *none*, following our *state-to-summary* converting method, the synthesized gold summary does not mention those slots. This behavior conforms to commonsense such that a summary generally does not include information not mentioned in the source text.

In a MultiWoZ dialogue, speakers often talk about multiple domains, so the synthesized summary should also mention the values from multiple domains. Given a multi-domain state, we split the state by different domains, and convert each single-domain partial state to a summary sentence. Then the resulting sentences are connected to a multiple-sentence summary. To be more natural, we paraphrased the common sentence prefix "The user is looking for" to "He is searching for" or "He looks for" for later utterances. For more examples, please refer to the Appendix Table 13.

3.4 Summary-to-state Converter

From a generated summary, a dialogue state is extracted by summary-to-state converter η. Based on the same template, the process is almost[2] the inverse of summary synthesis. We first split the whole summary into sentences from different domains. Domain-specific sentence prefix is used to identify which sentence is from which domain. The remaining process is to convert each single domain's one-sentence summary to a single-domain dialogue state and to finally merge them into one set of states. To convert a single-domain summary, slot values are extracted through string pattern matching via regular expressions based on the slot phrase templates from Section 3.3.

4 Experiments

4.1 Dataset

MultiWoZ (Budzianowski et al., 2018) is a large-scale English multi-domain task-oriented dialogue dataset. It contains 7 different domains, but as in Wu et al. (2019), we only use 5 out of them: train, hotel, restaurant, attraction, and taxi. Table 4 shows the number of dialogues for each domain in the training set of MultiWoZ 2.1. We evaluate DS2 on both MultiWoZ 2.0 and MultiWoZ 2.1, as most of the benchmark performances were reported on MultiWoZ 2.0.

[2]some slot-value entities include prepositions.

SAMSum (Gliwa et al., 2019) is a dialogue summarization dataset. We further pre-train T5-large (Raffel et al., 2020) more with SAMSum by using the code from Wu et al. (2021) before we fine-tune for DS2.

4.2 Evaluation

DST The main performance metric for our few-shot DST experiments is Joint Goal Accuracy (JGA). For each turn, only if the model's output dialogue state is exactly the same as the set of gold labels, we consider it correct (Balaraman et al., 2021). We report both all-domain JGA and per-domain JGA as in Wu et al. (2019) based on the evaluation setting that is described in the below Section 4.4. Slot accuracy is also computed for both active slots and none slots.

Dialogue Summarization In addition to the metrics for dialogue state prediction, we also use metrics to measure the quality of the intermediate dialogue summaries $\hat{\mathbf{z}}$. We measure BLEU-4 (Papineni et al., 2002) and ROUGE-4 (F1) (Lin, 2004) scores to evaluate how close a model-generated summary is to the synthesized gold summary. We also use ROUGE score to measure the performance of pre-training T5-large on the SAMSum corpus. The summarization performance are shown in Table 5.

4.3 Model

We mainly experiment with two pre-trained language models, T5-large and BART-large, as the summarization models of DS2. The pre-trained weights of T5-large from Raffel et al. (2020) are trained on mail and news data summarization. Hence, as mentioned above, we further pre-train the model with dialogue summarization.[3] To be specific, we pre-pend the prefix *Summarize this dialogue:* to $\mathbf{x}$ as done in the recent T0 (Sanh et al., 2021). We use BART-large that is already pre-trained on both XSum (Narayan et al., 2018) and SAMSum from Wu et al. (2021). In the ablation studies (Section 6.2), to compare the effectiveness of SAMSum pre-training, we use the original BART-large pre-trained on XSum (Lewis et al., 2020).

[3]The T5-large model we pre-trained on SAMSum corpus is released here: https://huggingface.co/jaynlp/t5-large-samsum

Model (*ver.* / mode)	Attraction			Hotel			Restaurant			Taxi			Train		
	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
TRADE (2.0 / CD)	35.8	57.5	63.1	19.7	37.4	41.4	42.4	55.7	60.0	63.8	66.5	70.1	59.8	69.2	71.1
DSTQA (2.0 / CD)	-	**70.4**	**71.6**	-	50.1	53.6	-	58.9	64.5	-	70.9	74.1	-	70.3	74.5
T5-DST (2.0 / CD)	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
CINS (2.0 / CT)	45.6	61.2	-	33.9	46.2	-	40.6	53.9	-	59.7	63.3	-	60.3	73.8	-
STARC (2.0 / CT)	40.3	65.3	66.2	**45.9**	**52.5**	**57.3**	51.6	60.4	64.6	72.5	75.3	79.6	65.6	74.1	75.0
TransferQA (2.0 / CT)	52.3	63.5	68.2	43.4	52.1	55.7	51.7	60.7	62.9	**75.4**	**79.2**	**80.3**	70.1	75.6	**79.0**
DS2 (2.0 / CD)	**65.26**	69.40	70.89	44.34	52.16	53.79	**58.94**	**64.12**	**64.65**	74.15	77.18	78.50	**74.21**	**76.96**	78.60
DS2 (2.0 / CT)	55.84	65.32	68.73	37.78	48.02	51.82	48.57	61.37	64.61	68.62	72.60	75.53	70.37	75.68	78.16
DS2 (2.0 / MD)	62.28	69.30	70.88	38.65	50.61	51.20	54.46	61.98	64.52	71.03	75.10	76.90	70.41	75.87	78.08
TransferQA (2.1 / CT)	50.25	60.92	64.28	32.46	39.02	41.99	47.12	59.16	62.24	71.12	74.47	76.07	69.01	73.17	75.46
DS2 (2.1 / CD)	**60.04**	**68.74**	**70.31**	**43.02**	**48.44**	**50.35**	**56.54**	**65.11**	**67.26**	**76.41**	**79.81**	**80.62**	**73.07**	**76.18**	77.00
DS2 (2.1 / CT)	53.60	64.44	66.90	36.17	46.96	48.29	48.36	63.96	66.82	68.84	76.82	77.23	67.96	75.55	**77.14**
DS2 (2.1 / MD)	56.33	66.39	67.14	38.22	47.75	48.34	50.19	63.22	64.45	71.87	77.10	79.01	69.87	75.55	76.36

Table 3: Per-domain few-shot (1-5-10%) results on MultiWOZ 2.0 and 2.1 (*ver.*). All of our **DS2** results are averaged over *3 runs (seeds)* and full results of each run are in the Appendix Tables 15,16. CD, CT, MD each refer to *Cross-Domain*, *Cross-Task*, *Multi-Domain* few-shot scenarios. We pre-trained TransferQA ourselves and fine-tuned it on *ver.* 2.1 to get the results, while all other results were taken from their respective papers. Note that we compare CD, CT, MD together as they all share the same test-set. Our proposed model **DS2** based on T5-large either achieves **SOTA** (bold) or competitive (underlined; ∼1.5-point difference) results in 2.0, and for 2.1 with the CD setting we outperform the SOTA model in 2.0 - TransferQA.

MultiWoZ 2.1	single-domain	multi-domain
Hotel	513	3381
Taxi	325	1654
Attraction	127	2717
Restaurant	1197	3813
Train	275	3103

Table 4: Number of dialogues for each domain in MultiWoZ 2.1 training set. Single-domain dialogues are subset of multi-domain dialogues.

Model	Rouge-1	Rouge-2	Rouge-L	# Params
PEGASUS (Zhang et al., 2020b)	50.50	27.23	49.32	568M
BART-large (Lewis et al., 2020)	51.74	26.46	48.72	406M
T5-large (Raffel et al., 2020)	**52.69**	**27.42**	**49.85**	770M

Table 5: Dialogue summarization results on SAMSum corpus (Gliwa et al., 2019). Both BART and PEGASUS numbers are taken from Wu et al. (2021), while for T5-large, we pretrained it using the code from Wu et al. (2021). Given such summarization results, we choose to use T5-large and BART-large.

4.4 Few-Shot Settings

There are three different scenarios for few-shot DST experiments:

- Cross-Domain (CD) (Wu et al., 2019)
- Cross-Task (CT) (Gao et al., 2020)
- Multi-Domain (MD) (Wu et al., 2020b)

For each setting, **1%, 5%, 10%**, or **100%** of training data is sampled to fine-tune a model. For all settings, we use the entire dev and test data for evaluation. As described in Section 4.1, we run each scenario for both MultiWoZ 2.0 and 2.1.

Cross-Domain CD was first explored by Wu et al. (2019) in MultiWoZ 2.0. In this setting, we consider the scenario of adapting a Dialogue System to a new target domain (e.g. taxi) while we have full training data for the source domains (e.g. restaurant, hotel, attraction, train). For this setting, we pre-train DS2 on all the source domains and then fine-tune the target domain. Note that during target-domain fine-tuning, as most of the dialogues are multi-domain (Table 4), we train DS2 to output summaries for **all domains** during the adaptation as well. During evaluation, only per-domain JGA is reported as in Wu et al. (2019).

Cross-Task CT was first explored for MultiWoZ by Gao et al. (2020) to demonstrate zero-shot DST performance. In our case, the difference with CD is that there is no source-domain pre-training and only target-domain fine-tuning is done. We measure per-domain JGA exactly as we do in CD.

Multi-Domain For MD experiments all domains are used to train a model. Every slot value is used for both summary synthesis and evaluation. Both JGA per-domain and total JGA are measured for multi-domain DST. We also evaluate full-shot training for multi-domain DST.

Model (*ver.*)	1%	5%	10%	100%
TRADE (2.0) (Wu et al., 2019)	11.74 (-)	32.41 (-)	37.42 (-)	48.62
TRADE + Self-supervision (2.0) (Wu et al., 2020b)	23.0 (-)	37.82 (-)	40.65 (-)	-
MinTL* (2.0) (Lin et al., 2020)	9.25 (2.33)	21.28 (1.94)	30.32 (2.14)	52.10
SOLOIST* (2.0) (Peng et al., 2021)	13.21 (1.97)	26.53 (1.62)	32.42 (1.13)	53.20
PPTOD* (2.0) (Su et al., 2021)	31.46 (0.41)	43.61 (0.42)	45.96 (0.66)	53.89
DS2 - T5 (2.0)	**36.15 (1.87)**	**45.14 (1.69)**	**47.61 (0.37)**	54.78
TRADE (2.1) (Wu et al., 2020b)	12.58 (-)	31.17 (-)	36.18 (-)	46.00
TRADE + Self-supervision (2.1) (Wu et al., 2020b)	21.90 (-)	35.13 (-)	38.12 (-)	-
DS2 - BART (2.1)	28.25 (0.98)	37.71 (1.05)	40.29 (0.29)	46.86
DS2 - T5 (2.1)	**33.76 (1.49)**	**44.20 (0.98)**	**45.38 (1.05)**	52.32

Table 6: Multi-domain Few-shot (1-5-10%) JGA evaluated on all domains jointly. *: taken from PPTOD (Su et al., 2021). Our models were run 3 times and full results are in Appendix Table 17.

4.5 Baselines

All baseline results are only reported on MultiWoZ 2.0, and we additionally experimented with TransferQA on 2.1 as it was the best model.

TRADE (CD, MD) Wu et al. (2019) utilizes copy mechanism and slot & domain embeddings for transferability. Meanwhile, Wu et al. (2020b) applies self-supervision to improve zero-shot and few-shot CD& MD performances of TRADE.

T5-DST (CD) Lin et al. (2021b) prompts a T5 model with slot descriptions for few-shot DST.

STARC (CT) Gao et al. (2020) asks natural language questions separately to two different instances of RoBERTa-Large (Liu et al., 2019) for categorical and non-categorical slots.

TransferQA (CT) Lin et al. (2021a) asks natural language questions to a single T5-large model that is pre-trained to predict none values properly. As the original authors did not release their pre-trained version we release our own using their code[4].

CINS (CT) Mi et al. (2021) prompts a T5-base with slot descriptions for few-shot DST.

DSTQA (CD) Zhou and Small (2019) performs DST via question answering over ontology graph.

PPTOD (MD) (Su et al., 2021) prompts a PLM pre-trained on various TOD task and data with natural language instructions.

5 Result

5.1 Few-shot: per-domain

Table 3 shows the result of the few shot performance of DS2 compared to the baselines in three different settings described in Section 4.4. To compare with previous studies, we also evaluate our model on MultiWOZ 2.0 version. In *ver.* 2.0, Lin et al. (2021a); Gao et al. (2020) show that even without cross-domain pre-training CT models can outperform CD ones. We believe that this is can be attributed to the usage of large pre-trained language models like T5-large (∼770M parameters). When we use the same-sized model, we outperform all other CT models in **1% setting** (30∼50 dialogues) for 3 domains and achieve very competitive results in the 2 other domains. When evaluating *ver.* 2.0's SOTA model TransferQA on *ver.* 2.1, we can, in fact, see that DS2 significantly outperforms it in all domains. We show slot accuracy and other metrics in Appendix Table 12.

5.2 Few shot: all-domain

In Table 6, we also show all-domain few-shot performance of DS2 in the MD setting compared to previous works. From the table, it is clear that for all 1%, 5%, 10% few-shot adaptation settings, DS2 achieves SOTA performance in both MultiWOZ 2.0 and 2.1. It is also worth noting that we outperform PPTOD which not only uses T5-Large as well but also pre-trains their model on various TOD tasks and datasets. In addition, in the table, we also report the full-shot performance of DS2

[4]TransferQA pre-trained on the QA data: https://huggingface.co/jaynlp/t5-large-transferqa

Error type	**Hallucination** : The model generates unmentioned information.
Pattern	The user is looking for a train from ____ to ____ on ____, which leaves at ____.
Summary	The user is looking for a train for 7 people from broxbourne to cambridge on wednesday, which arrives at 11:30.
Gold	The user is looking for a train from broxbourne to cambridge on wednesday, which leaves at 11:30.
Error type	**Missing slot** : The model omits expected slot.
Pattern	The user is looking for a train from ____ on ____, which leaves at ____.
Summary	The user is looking for a train from peterborough on friday.
Gold	The user is looking for a train from peterborough on friday, which leaves at 16:00.
Error type	**Wrong slot** : The model mismatches slot template of the given information.
Pattern	The user is looking for a train for ____ people from ____ to ____ on ____, which leaves at ____.
Summary	The user is looking for a train for 2 people from bishops stortford to cambridge on thursday, which arrives by 18:30.
Gold	The user is looking for a train for 2 people from bishops stortford to cambridge on thursday, which leaves at 18:30.

Table 7: Three common error types of DS2. Dialogue id's of examples: MUL0603, SNG0271, PMUL4126.

Model	Inference Time Complexity
DSTReader (Gao et al., 2019)	$O(k\tau)$
TRADE (Wu et al., 2019)	$O(k\tau)$
COMER (Ren et al., 2019)	$O(k\tau)$
SOM-DST (Kim et al., 2020b)	$O(k\tau)$
T5-DST (Lin et al., 2021b)	$O(k\tau)$
STARC (Gao et al., 2020)	$O(k\tau)$
TransferQA (Lin et al., 2021a)	$O(k\tau)$
CINS (Mi et al., 2021)	$O(k\tau)$
PPTOD (Su et al., 2021)	$O(k+\tau)$
NADST (Le et al., 2019)	$O(k+\tau)$
DS2 (Ours)	$O(k+\tau)$

Table 8: Worst-case inference time complexity adapted from Ren et al. (2019); Kim et al. (2020b). k for number of slots and τ for model inference time.

which is 54.78 (2.0) & 52.32 (2.1): relatively strong numbers considering that we did not put in any task-specific engineering as in Heck et al. (2020); Yu et al. (2021).

6 Analysis

6.1 Time Complexity

Our method DS2 is efficient in terms of inference speed. Table 8 shows inference time complexity. k and τ each denote the number of slots and the model inference time. The numbers for other models are modified from Ren et al. (2019) and Kim et al. (2020b). Other models, except for the bottom three models including DS2, has $O(k\tau)$ time complexity. For instance, QA-based models should ask a question for every potential slot in the given domains, so it requires k times more model inference. On the other hand, DS2 only needs to run the PLM once for summary generation. After that, *summary-to-state* pattern matching takes $O(k)$ time.

Training Options	JGA (std)
DS2 (BART-large)	28.3 (0.98)
- *SAMSum pre-training*	25.5 (1.46)
- *dontcare concat*	27.1 (0.97)
- *paraphrasing*	23.6 (0.71)
- *paraphrasing & dontcare concat*	23.5 (1.86)
- *summary naturalness*	13.1 (0.45)

Table 9: Effects of SAMSum pre-training and template naturalness. Each row subtracts a module from the best setting of DS2. We show 3-run validation JGA for MD 1% few shot training of BART-large on 2.1.

6.2 Ablation Study

In this section, we analyze the key components that to our model's success.

Dialogue Summary Pre-training As mentioned in Section 4.3, we further pre-train T5-large on the SAMSum corpus. The second row of Table 9 shows what led to this decision. We observe that pre-training SAMSum had a large effect on BART-large (∼440M parameters). In addition, we include the evaluation results on SAMSum in the Table 5; overall T5-large performed better than other models.

Summary Naturalness As mentioned in the last paragraph of Section 3.2, guiding the generated summaries to conform to our synthetic templates is not a trivial task, and we hypothesized that the naturalness of these templates is key to successful performance. To answer this question, we conducted an ablation study on the *state-to-summary* converter in Table 9. The details of each *state-to-summary* converter is shown in the Appendix Table 14. In short, 1) *paraphrasing* refers to whether we allow multiple prefixes and pronouns when

synthesizing summary labels, 2) *dontcare concat* is whether we use single or two sentences when adding *dontcare* related phrases, and 3) finally, *summary naturalness* is whether we use human-like language and grammar when making the summary. From the table, we can clearly see a significant performance drop when we disable *summary naturalness*. Meanwhile, disabling *paraphrasing* also had a non-negligible impact on the JGA, but *dontcare concat* had only a minor decrease in performance. Therefore, we conjecture that because we provide much more natural labels to the model, we can outperform PPTOD in Table 6.

6.3 Error Analysis

Table 7 shows failure cases of DS2 summary model. Correctly predicted slot values are highlighted with blue color, while wrong ones are red. We report three categories of typical failures: "hallucination", "missing slot", and "wrong slot". Shuster et al. (2021), Durmus et al. (2020) reports "Hallucination" is a phenomenon in which a model generates unmentioned information in original dialogue. "Missing slot" is the most commonly observed case where the predicted summary omits information on a required slot. Similar failures also happen at the domain-level. The third type is named "wrong slot", where the model confuses two slots with same data types. For example, values for both "arrive-by" and "leaves-at" have same formats, so the model often fails to discriminating them.

7 Cost of Template Engineering

For MultiWoZ, we devised templates for all 30 slots in the 5 domains that were used. Based on names of the slots, we wrote a `state-to-summary` function that generates a natural phrase along with the slot value with prefix templates. The `summary-to-state` parsing functions were written using regular expressions based on the rules we implemented for template generation. Overall, this process took approximately one week for one expert to finish. We believe that this is a much lower cost compared to full DST data design and collection efforts. Applying to a new domain may take even lower costs by using our code-base. Appendix Section A.4 describes this process in detail.

8 Limitations

In this section, we discuss several limitations of this work. First, applying our model to a new domain requires a new summary template. Since DS2 performance is sensitive to the quality of the template as shown in the ablation study, considerable amount of knowledge on both domain and NLP is desired. However, following the guide in Section A.4 would take less than one week for a researcher, which costs much less than collecting full DST data. Second, DS2 is not capable of zero-shot inference because it should learn the template, at least from a few samples. Third, regular expression pattern matching may fail during the state extraction. There is no guarantee for the model output to fit in the template. The matching may still fail for a correctly formatted summary if a value entity contains template-like patterns. Using a neural network-based converter might easily solve this problem. Fourth, there is still room for improvement using DST-specific engineering (span matching or ontology searching as in TripPy (Heck et al., 2020)). Finally, output summary length is bounded by the PLM's maximum sequence length, so DS2 might fail when we have too many slot values. We leave these for future investigation.

9 Conclusion

This work tackles the few-shot DST problem by reformulating it into dialogue summarization. The strategy is to minimize the pre-train and fine-tune discrepancy by adapting a Pre-trained Language Model (PLM) to a more familiar task: summarization. Hence, instead of forcing the model to learn a completely new task like DST, we provide rule-based summary templates from dialogue states. We guide the summarization to conform to such templates, and utilize heuristic dialogue state extraction from the generated summaries. The experimental results show that our model DS2 outperforms baselines for few-shot DST on MultiWoZ in both cross-domain and multi-domain settings. In addition, DS2 significantly reduces inference time complexity compared to existing QA-based methods. We also observed that naturalness of the template was very important.

Acknowledgements

We would like to thank Whakyeong Seo and Wansoo Kim of Riiid very much for their gracious support on designing the figures and helping us scale up our experiments to the Google Cloud Platform. We would also like to thank Zhaonjiang Lin for the helpful discussions.

References

Vevake Balaraman, Seyedmostafa Sheikhalishahi, and Bernardo Magnini. 2021. Recent neural methods on dialogue state tracking for task-oriented dialogue systems: A survey. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 239–251.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Ultes Stefan, Ramadan Osman, and Milica Gašić. 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 (EMNLP)*.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica 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*, pages 122–132.

Jiaao Chen and Diyi Yang. 2020. Multi-view sequence-to-sequence models with conversational structure for abstractive dialogue summarization. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4106–4118.

Lu Chen, Boer Lv, Chunxin Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *AAAI 2020*.

Yulong Chen, Yang Liu, and Yue Zhang. 2021. Dialogsum challenge: Summarizing real-life scenario dialogues. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 308–313.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Esin Durmus, He He, and Mona Diab. 2020. Feqa: A question answering evaluation framework for faithfulness assessment in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5055–5070.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 422–428, Marseille, France. European Language Resources Association.

Alexander Fabbri, Faiaz Rahman, Imad Rizvi, Borui Wang, Haoran Li, Yashar Mehdad, and Dragomir Radev. 2021. ConvoSumm: Conversation summarization benchmark and improved abstractive summarization with argument mining. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6866–6880, Online. Association for Computational Linguistics.

Xiachong Feng, Xiaocheng Feng, Libo Qin, Bing Qin, and Ting Liu. 2021. Language model as an annotator: Exploring DialoGPT for dialogue summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1479–1491, Online. Association for Computational Linguistics.

Shuyang Gao, Sanchit Agarwal, Di Jin, Tagyoung Chung, and Dilek Hakkani-Tur. 2020. From machine reading comprehension to dialogue state tracking: Bridging the gap. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 79–89.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273.

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 (Volume 1: Long Papers)*, pages 3816–3830, Online. Association for Computational Linguistics.

Bogdan Gliwa, Iwona Mochol, Maciej Biesek, and Aleksander Wawer. 2019. Samsum corpus: A human-annotated dialogue dataset for abstractive summarization. *EMNLP-IJCNLP 2019*, page 70.

Ting Han, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang. 2021. Multiwoz 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 206–218. Springer.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the*

Special Interest Group on Discourse and Dialogue, pages 35–44.

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*, volume 33, pages 20179–20191. Curran Associates, Inc.

John F Kelley. 1984. An iterative design methodology for user-friendly natural language office information applications. *ACM Transactions on Information Systems (TOIS)*, 2(1):26–41.

Muhammad Khalifa, Miguel Ballesteros, and Kathleen Mckeown. 2021. A bag of tricks for dialogue summarization. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8014–8022.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020a. Efficient dialogue state tracking by selectively overwriting memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582, Online. Association for Computational Linguistics.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020b. Efficient dialogue state tracking by selectively overwriting memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582.

Hung Le, Richard Socher, and Steven CH Hoi. 2019. Non-autoregressive dialog state tracking. In *International Conference on Learning Representations*.

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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880.

Shiyang Li, Semih Yavuz, Kazuma Hashimoto, Jia Li, Tong Niu, Nazneen Rajani, Xifeng Yan, Yingbo Zhou, and Caiming Xiong. 2021a. Coco: Controllable counterfactuals for evaluating dialogue state trackers. In *International Conference on Learning Representations*.

Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, and Julian McAuley. 2021b. 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*, pages 1063–1074.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Zhenpeng Zhou, Paul A Crook, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, et al. 2021a. Zero-shot dialogue state tracking via cross-task transfer. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7890–7900.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul A Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Andrea Madotto, Eunjoon Cho, and Rajen Subba. 2021b. Leveraging slot descriptions for zero-shot cross-domain dialogue statetracking. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5640–5648.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020. Mintl: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405.

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

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Andrea Madotto, Zhaojiang Lin, Genta Indra Winata, and Pascale Fung. 2021. Few-shot bot: Prompt-based learning for dialogue systems. *arXiv preprint arXiv:2110.08118*.

Shikib Mehri, Mihail Eric, and Dilek Hakkani-Tur. 2020. Dialoglue: A natural language understanding benchmark for task-oriented dialogue. *arXiv preprint arXiv:2009.13570*.

Fei Mi, Yitong Li, Yasheng Wang, Xin Jiang, and Qun Liu. 2021. Cins: Comprehensive instruction for few-shot learning in task-oriented dialog systems. *arXiv preprint arXiv:2109.04645*.

Shashi Narayan, Shay B Cohen, and Mirella Lapata. 2018. Don't give me the details, just the summary! topic-aware convolutional neural networks for extreme summarization. In *Proceedings of the 2018*

Conference on Empirical Methods in Natural Language Processing, pages 1797–1807.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Sungjoon Park, Jihyung Moon, Sungdong Kim, Won Ik Cho, Ji Yoon Han, Jangwon Park, Chisung Song, Junseong Kim, Youngsook Song, Taehwan Oh, et al. 2021. Klue: Korean language understanding evaluation. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*.

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. *Transactions of the Association for Computational Linguistics*, 9:807–824.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21:1–67.

Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. 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)*, pages 1876–1885.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

Timo Schick and Hinrich Schütze. 2021a. 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*, pages 255–269.

Timo Schick and Hinrich Schütze. 2021b. Few-shot text generation with natural language instructions. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 390–402, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Timo Schick and Hinrich Schütze. 2021c. It's not just size that matters: Small language models are also few-shot learners. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2339–2352.

Richard Shin, Christopher H. Lin, Sam Thomson, Charles Chen, Subhro Roy, Emmanouil Antonios Platanios, Adam Pauls, Dan Klein, Jason Eisner, and Benjamin Van Durme. 2021. Constrained language models yield few-shot semantic parsers. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. 2021. Retrieval augmentation reduces hallucination in conversation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 3784–3803, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2021. Multi-task pre-training for plug-and-play task-oriented dialogue system. *arXiv preprint arXiv:2109.14739*.

Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine*, 35(4):121–124.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

Chien-Sheng Wu, Steven CH 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)*, pages 917–929.

Chien-Sheng Wu, Steven CH Hoi, and Caiming Xiong. 2020b. Improving limited labeled dialogue state tracking with self-supervision. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, pages 4462–4472.

Chien-Sheng Wu, Linqing Liu, Wenhao Liu, Pontus Stenetorp, and Caiming Xiong. 2021. Controllable abstractive dialogue summarization with sketch supervision. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 5108–5122, Online. 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 Annual Meeting of the Association for Computational Linguistics*, pages 808–819.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2021. Multiwoz 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. *arXiv preprint arXiv:2104.00773*.

Tao Yu, Rui Zhang, Alex Polozov, Christopher Meek, and Ahmed Hassan Awadallah. 2021. Score: Pre-training for context representation in conversational semantic parsing. In *International Conference on Learning Representations*.

Xiaoxue Zang, Abhinav Rastogi, Srinivas Sunkara, Raghav Gupta, Jianguo Zhang, and Jindong Chen. 2020. Multiwoz 2.2: A dialogue dataset with additional annotation corrections and state tracking baselines. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 109–117.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, S Yu Philip, Richard Socher, and Caiming Xiong. 2020a. 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*, pages 154–167.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter Liu. 2020b. Pegasus: Pre-training with extracted gap-sentences for abstractive summarization. In *International Conference on Machine Learning*, pages 11328–11339. PMLR.

Shiyue Zhang, Asli Celikyilmaz, Jianfeng Gao, and Mohit Bansal. 2021. Emailsum: Abstractive email thread summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6895–6909.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020c. Task-oriented dialog systems that consider multiple appropriate responses under the same context. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9604–9611.

Ming Zhong, Da Yin, Tao Yu, Ahmad Zaidi, Mutethia Mutuma, Rahul Jha, Ahmed Hassan, Asli Celikyilmaz, Yang Liu, Xipeng Qiu, et al. 2021. Qmsum: A new benchmark for query-based multi-domain meeting summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5905–5921.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *ArXiv*, abs/1911.06192.

Chenguang Zhu, Yang Liu, Jie Mei, and Michael Zeng. 2021. Mediasum: A large-scale media interview dataset for dialogue summarization. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5927–5934.

A Appendix

A.1 *State-to-summary* Ablation Details

DST Performance improvement is driven by naturalness of summary template used for summary generation. To provide understanding of converting options we explored, Table 13 shows an example sentence for each domain. In the table, all kinds of slots are introduced with example values. Example summary is constructed combining given slot values with corresponding slot templates as Table 2. Last row shows the case of multi-domain dialogue. Sentence of each domain is concatenated by a conjunction 'Also', in random order for balanced training.

Table 14 shows difference between several converting options, which performance is compared in the previous Table 9. Unnatural converter was proposed to make a prediction without domain knowledge, so it generates slot names itself, while other converters do not generate names of domain or slot name.

Other ablation options are compared to each other under fair condition. The second option from the bottom was our initial idea. All domain's summary sentence shares same sentence prefix and the summary for *dontcare* value was handled separately due to its quite different semantics from other values. Paraphrasing seems to be effective, and we assume that is because the model was trained to avoid repetition of same phrase during their generative pre-train tasks. Concatenating *dontcare* sentence is proposed from the idea that if the dialogue has several domain and more then one domain contains *dontcare* slot, the number of sentence might be too large.

A.2 Experiment Details

We used cloud computing instances with NVIDIA Tesla A100 GPU for pre-training and fine-tuning t5-large model, and on-premise computing environment with NVIDIA GeForce RTX 2080 Ti for training BART-large model. Except pre-training task of Lin et al. (2021a) model as MultiWOZ 2.1 baseline implementation, we didn't use any distributed data parallel setting, so multiple GPUs are not used to train our DS2 model.

All experiments for DS2 used `pytorch-lightning` and `huggingface` libraries. Requirements for other software used are specified in the `requirements.txt` in the accompanying code. Training epoch is fixed with 100, while early stopping callback was enabled with patient 10 on validation joint goal accuracy metric, so most of the final number of epochs are within 10 30 epochs. Training batch size was 2 for T5-large, and 1 for BART. We used greedy search on transformer model's auto regressive language generation for speed, by setting number of beams parameter of pytorch model as 1. Accumulating gradient batches options are available in pytorch lightning trainer module, we set accumulate grad batches options to 1, 5, 10, 100 for 1%, 5%, 10%, 100% few shot learning. MultiWOZ dataset provides train, dev, test splits, so we used the given splits.

T5-Large	CT	CD	MD
1%	8∼10	8∼10	14∼16
5%	10∼12	10∼12	15∼17
10%	12∼14	12∼14	16∼18
100% / Pretraining	-	30∼40	30∼40

Table 10: Estimated train/validation time (GPU hours) through virtual resource usage record. We used NVIDIA Tesla A100 through Google Cloud Platform.

A.3 Dataset and Model

Table 4 shows number of dialogues in MultiWOZ 2.1 datasets. Single-domain dialogue is defined by a dialogue annotated with only one domain. Since appearance of unrelated domain information on dialogue may harm summarization's nature due to difference from dialogue state information to original text, single domain dialogues are the ideal requirements for cross-task setting. Lack of these single-domain dialogue as shown in table leads us to focus on cross domain setting, which can be performed naturally.

Information we used on model selection is on Table 5. Comparing to previous study's reported performance of summarization on SAMSum corpus, T5-large does summarization well. Since larger models like T5-3B is harder to train with limited GPU resource, and previous work, Lin et al. (2021a) was also evaluated by T5-large, we selected T5-large as base summarization model. While there is no public T5-large model weight trained with SAMSum data, we pretrained T5-large by using the code from CODS[5].

[5] https://github.com/salesforce/ConvSumm/tree/master/CODS

In addition to T5-large, we also did many experiments using BART-large because smaller model weights of BART allows to be trained in single 2080Ti GPU, which costs much lower. For the comparative ablation experiment introduced in Table 9, we used off-the-shelf weights for both SAMSum-unseen[6] and SAMSum-pretrained[7] weights.

A.4 Guide for applying DS2 to a new domain

The most plausible scenario of reusing our code is to apply it to a new dialogue domain. For that purpose, it is sufficient to rewrite the heuristic converters between dialogue states and summaries. It might take several hours for a Python developer to implement.

As explained in 3.3, our converting method is built in a hierarchical manner. Therefore, following the original design is the best strategy to add code for a new domain.

1. Define natural language description for new domain.
    - Define natural language templates for each domain and slot. e.g) in case of the slot "hotel-name", we can create a summary template sentence "The user is looking for a place to stay called x."
    - Define natural language description for each slot to cover don't care scenario. e.g), in case of the slot "hotel-name", the summary sentence can be "The user is looking for a hotel and he does not care about the name".
2. Replace the code with your expression.
    - We explicitly defined natural language expressions with python dictionary at the top of the converter python script. Inject your expression into the corresponding dictionary.
3. Modify converter code if you want to control plural form, article, space, or quotation marks. The final state-to-summary converter is written as Code 1.
4. Write a summary-to-state converter for the domain according to the intended expression as in Code 2.

[6] https://huggingface.co/facebook/bart-large-xsum

[7] https://huggingface.co/Salesforce/bart-large-xsum-samsum

```
def hotel_state_to_sum(ds: dict, either: callable, is_one_sentence: bool, idx: int,
    wo_para: bool) -> str:
    first_sentence = get_first_sentence(ds=ds, domain="hotel", either=either,
    except_keys={"hotel-parking", "hotel-internet"}, idx=idx, wo_para=wo_para)

    second_sentence = get_dontcare_sentence(
        ds,
        domain="hotel",
        either=either,
        is_one_sentence=is_one_sentence,
        wo_para=wo_para
    )

    res = first_sentence + second_sentence + "."
    return res
```

Code 1: State-to-summary converter in Python code for the domain 'hotel'.

```
import re
...

def hotel_sum_to_state(summ: str, is_one_sentence: bool) -> dict:
    sentences = re.split("|".join(COMMON_PHRASES), summ)
    summary = [sentence for sentence in sentences if DOMAIN_PHRASE_IN_SENTENCE["
    hotel"] in sentence]
    if not summary:
        return {}
    summary = summary[0]
    slot_to_prefix = {
        "hotel-type": " which is a ",
        "hotel-name": " called ",
        "hotel-stars": " ranked ",
        "hotel-pricerange": " with a",
        "hotel-area": " located in the ",
        "hotel-book people": r" for \d+ p",
        "hotel-book day": " on ",
        "hotel-book stay": r" for \d+ d",
        "hotel-parking": [" has no p", " has p"],
        "hotel-internet": [" has no i", " has i"],
    }
    res = {}

    dontcare_sentence = summary
    if not is_one_sentence:
        summary = summary.split('.')[0]

    for slot, prefix in slot_to_prefix.items():
        if type(prefix) == str:
            matches = [re.search(prefix, summary)]
        else:
            matches = [re.search(p, summary) for p in prefix]
        for match in matches:
            if match:
                start_idx = match.span()[-1]
                if slot in {"hotel-book people", "hotel-book stay"}:
                    start_idx -= 3
                elif slot == "hotel-pricerange":
                    start_idx += 2 if summary[start_idx:].startswith("n") else 1

                _summary = summary[start_idx:]

                value = re.split(
                    " The | Also, | which | called | ranked | during | located in
    the | for | on | and | with a| people| person| price| star| day",
                    _summary,
                )[0]

                if slot in ["hotel-internet", "hotel-parking"]:
                    value = "no" if " no " in match.group() else "yes"

                res[slot] = value.replace(",", "").replace(".", "")

    res.update(get_dontcare_values(dontcare_sentence, domain="hotel"))

    return res
```

Code 2: Summary-to-state converter in Python code for the domain 'hotel'.

ACL Reproducibility Guideline	
For all reported experimental results	
A clear description of the mathematical setting, algorithm, and/or model	O
A link to (anonymized, for submission) downloadable source code, with specification of all dependencies, including external libraries	O
A description of the computing infrastructure used	O
The average runtime for each model or algorithm, or estimated energy cost	X
The number of parameters in each model	O
Corresponding validation performance for each reported test result	X
A clear definition of the specific evaluation measure or statistics used to report results	O
For all results involving multiple experiments, such as hyperparameter search	
The exact number of training and evaluation runs	O
The bounds for each hyperparameter	Not tuned
The hyperparameter configurations for best-performing models	Not tuned
The method of choosing hyperparameter values (e.g., manual tuning, uniform sampling, etc.) and the criterion used to select among them (e.g., accuracy)	Not tuned
Summary statistics of the results (e.g., mean, variance, error bars, etc.)	O
For all datasets used	
Relevant statistics such as number of examples and label distributions	O
Details of train/validation/test splits	O
An explanation of any data that were excluded, and all pre-processing steps	O
For natural language data, the name of the language(s)	O
A link to a downloadable version of the dataset or simulation environment	O
For new data collected, a complete description of the data collection process, such as ownership / licensing, informed consent, instructions to annotators and methods for quality control	X

Table 11: Reproducibility Checklist. We do not do extensive hyper-parameter tuning for our models.

T5-Large Cross domain	JGA	BLEU	Slot True Acc	Slot None Acc	Rouge-4
taxi 1%	0.764 (0.009)	0.812 (0.008)	0.729 (0.021)	0.958 (0.003)	0.797 (0.007)
taxi 5%	0.798 (0.010)	0.834 (0.008)	0.769 (0.004)	0.971 (0.005)	0.820 (0.003)
taxi 10%	0.806 (0.004)	0.839 (0.002)	0.779 (0.005)	0.973 (0.004)	0.823 (0.003)
hotel 1%	0.430 (0.020)	0.796 (0.008)	0.810 (0.016)	0.939 (0.000)	0.785 (0.009)
hotel 5%	0.484 (0.008)	0.830 (0.002)	0.839 (0.005)	0.955 (0.003)	0.823 (0.002)
hotel 10%	0.504 (0.011)	0.836 (0.005)	0.849 (0.011)	0.957 (0.004)	0.830 (0.004)
train 1%	0.731 (0.008)	0.828 (0.006)	0.906 (0.004)	0.972 (0.005)	0.814 (0.006)
train 5%	0.762 (0.004)	0.860 (0.000)	0.917 (0.003)	0.976 (0.003)	0.843 (0.000)
train 10%	0.770 (0.005)	0.863 (0.001)	0.922 (0.002)	0.977 (0.003)	0.846 (0.001)
attraction 1%	0.600 (0.016)	0.793 (0.006)	0.761 (0.009)	0.894 (0.013)	0.773 (0.006)
attraction 5%	0.687 (0.001)	0.825 (0.006)	0.840 (0.007)	0.909 (0.006)	0.803 (0.006)
attraction 10%	0.703 (0.004)	0.832 (0.002)	0.837 (0.002)	0.927 (0.005)	0.811 (0.002)
restaurant 1%	0.565 (0.031)	0.811 (0.006)	0.866 (0.037)	0.941 (0.014)	0.799 (0.007)
restaurant 5%	0.651 (0.004)	0.848 (0.001)	0.907 (0.005)	0.960 (0.001)	0.833 (0.001)
restaurant 10%	0.673 (0.020)	0.855 (0.004)	0.910 (0.010)	0.962 (0.006)	0.841 (0.004)

Table 12: Evaluation metrics for summary generation quality and slot prediction accuracy. Slot true accuracy means correctness rate for slots with existing value. Slot none accuracy is the metric for predict slots with none value as none. All of the value is mean (standard deviation) of three few shot trials

Domain	Example Dialogue State	Example summary
Taxi	taxi-departure: *london station* taxi-destination: *Incheon airport* taxi-arriveby: *12:30* taxi-leaveat: *02:45*	The user is looking for a taxi from *london station* to *Incheon airport*, which leaves at *02:45* and arrives by *12:30*.
Train	train-departure: *norwich* train-destination: *cambridge* train-arriveby: *19:45* train-book people: *3* train-leaveat: *11:21* train-day: *monday*	The user is looking for a train for *3* people from *norwich* to *cambridge* on *monday*, which leaves at *11:21* and arrives by *19:45*.
Hotel	hotel-type: *hotel* hotel-name: *Intercontinental* hotel-stars: *3* hotel-pricerange: *cheap* hotel-area: *east* hotel-book people: *6* hotel-book day: *saturday* hotel-book stay: *3* hotel-parking: *yes* hotel-internet: *no*	The user is looking for a place to stay which is a *hotel* called *Intercontinental* ranked *3* stars with a *cheap* price located in the *east* for *6* people on *saturday* for *3* days, which *has parking* and *has no internet*.
Attraction	attraction-area: *cambridge* attraction-name: *nusha* attraction-type: *entertainment*	The user is looking for an attraction which is an *entertainment* called *nusha* located in the *cambridge*.
Restaurant	restaurant-book day: *tuesday* restaurant-book people: *6* restaurant-book time: *12:00* restaurant-name: *meze bar* restaurant-pricerange: *cheap* restaurant-area: *south* restaurant-food: *seafood*	The user is looking for a restaurant called *meze bar* located in the *south* with a *cheap* price for *6* people on *tuesday* at *12:00*, which serves *seafood*.
Multiple domain	restaurant-book day: *tuesday* restaurant-book time: *12:00* restaurant-name: *meze bar* train-departure: *london station* train-destination: *Incheon airport* train-book people: *3* hotel-type: *guesthouse* hotel-name: *Intercontinental* hotel-stars: *3*	The user is looking for a train for *3* people from *london station* to *Incheon airport*. Also, he is searching for a restaurant called *meze bar* on *tuesday* at *12:00*. Also, he looks for a place to stay which is a *guesthouse* called *Intercontinental* ranked *3* stars.

Table 13: Example for summary template of each domain.

Sample Dialogue State	**hotel-area: dontcare** **hotel-pricerange: moderate** **hotel-internet: yes** **hotel-type: guesthouse**	**train-book people: 3** **train-leaveat: 10:30** **train-destination: cambridge** **train-day: tuesday** **train-departure: kings lynn**
Converter	**Example Summary**	
Natural Summary (DS2)	The user is looking for a place to stay which is a guesthouse with a moderate price, which has internet, and he does not care about the location. Also, he is searching for a train for 3 people from kings lynn to cambridge on tuesday, which leaves at 10:30	
Without paraphrasing repeated prefix (- *paraphrasing*)	The user is looking for a place to stay which is a guesthouse with a moderate price, which has internet, and the user does not care about the location. Also, the user is looking for is looking for a train for 3 people from kings lynn to cambridge on tuesday, which leaves at 10:30.	
Without concatenating don't care sentence (- *dontcare concat*)	The user is looking for a place to stay which is a guesthouse with a moderate price, which has internet. He does not care about the location. Also, he is searching for a train for 3 people from kings lynn to cambridge on tuesday, which leaves at 10:30.	
Without both paraphrasing, concatenating (- *paraphrasing* & *dontcare concat*)	The user is looking for a place to stay which is a guesthouse with a moderate price, which has internet. The user does not care about the location. Also, the user is looking for a train for 3 people from kings lynn to cambridge on tuesday, which leaves at 10:30.	
Unnatural Summary (- *summary naturalness*)	The user wants dontcare as area of hotel, moderate as pricerange of hotel, yes as internet of hotel, guesthouse as type of hotel, 3 as book people of train, 10:30 as leaveat of train, cambridge as destination of train, tuesday as day of train, kings lynn as departure of train.	

Table 14: Dialogue states from PMUL3853.json of MultiWOZ 2.1 and converted summary by using various converter options mentioned in Section 6.2. Differences by each converter options are pointed to blue text color.

T5 Large		Attraction			Hotel			Restaurant			Taxi			Train		
ver. & mode		1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
DS2 - 2.0 - CD	Run 1 (seed 11)	65.79	69.23	73.34	44.66	52.09	53.56	59.63	65.23	66.33	73.94	77.42	78.52	75.05	75.11	77.58
	Run 2 (seed 23)	65.76	70.48	70.35	43.82	53.37	54.06	57.52	63.02	63.53	74.26	76.52	77.87	72.40	79.31	80.21
	Run 3 (seed 47)	64.24	68.49	68.97	44.54	51.03	53.75	59.66	64.10	64.10	74.26	77.61	79.10	75.16	76.45	78.00
	Mean	65.26	69.40	70.89	44.34	52.16	53.79	58.94	64.12	64.65	74.15	77.18	78.50	74.20	76.96	78.60
	(Std.Dev)	(0.89)	(1.01)	(2.23)	(0.45)	(1.17)	(0.25)	(1.23)	(1.11)	(1.48)	(0.18)	(0.58)	(0.62)	(1.56)	(2.15)	(1.41)
DS2 - 2.0 - CT	Run 1 (seed 11)	56.82	66.08	70.71	39.64	48.88	51.31	50.49	61.09	65.11	68.77	72.32	75.81	70.24	75.08	79.05
	Run 2 (seed 23)	56.01	65.11	68.62	38.14	47.03	51.44	44.54	62.13	65.11	67.81	72.84	75.81	68.74	77.92	78.84
	Run 3 (seed 47)	54.69	64.76	66.85	35.55	48.16	52.72	50.67	60.88	63.62	69.29	72.65	74.97	72.13	74.03	76.58
	Mean	55.84	65.32	68.73	37.78	48.02	51.82	48.57	61.37	64.61	68.62	72.60	75.53	70.37	75.68	78.16
	(Std.Dev)	(1.08)	(0.68)	(1.93)	(2.07)	(0.93)	(0.78)	(3.49)	(0.67)	(0.86)	(0.75)	(0.26)	(0.48)	(1.70)	(2.01)	(1.37)
DS2 - 2.0 - MD	Run 1 (seed 11)	63.70	71.03	70.32	39.54	51.59	51.12	52.60	62.46	64.75	70.19	75.68	76.90	69.98	77.42	78.39
	Run 2 (seed 23)	61.93	68.62	70.93	42.17	52.15	53.84	55.40	62.13	65.14	72.00	75.29	77.16	71.27	75.37	76.24
	Run 3 (seed 47)	61.22	68.26	71.38	34.24	48.10	48.63	55.37	61.36	63.68	70.90	74.32	76.65	69.98	74.82	79.60
	Mean	62.28	69.30	70.88	38.65	50.61	51.20	54.46	61.98	64.52	71.03	75.10	76.90	70.41	75.87	78.08
	(Std.Dev)	(1.28)	(1.51)	(0.53)	(4.04)	(2.19)	(2.61)	(1.61)	(0.56)	(0.76)	(0.91)	(0.70)	(0.26)	(0.74)	(1.37)	(1.70)
DS2 - 2.1 - CD	Run 1 (seed 11)	57.88	68.94	70.45	45.44	47.82	49.34	59.04	65.41	68.62	75.68	80.19	81.23	71.92	76.00	77.37
	Run 2 (seed 23)	61.58	68.68	69.74	42.95	48.00	51.81	58.41	65.44	68.68	75.87	78.39	80.32	73.87	75.79	77.37
	Run 3 (seed 47)	60.68	68.59	70.74	40.67	49.50	49.91	52.16	64.48	64.48	77.68	80.84	80.32	73.42	76.76	76.26
	Mean	60.04	68.74	70.31	43.02	48.44	50.35	56.54	65.11	67.26	76.41	79.81	80.62	73.07	76.18	77.00
	(Std.Dev)	(1.93)	(0.18)	(0.51)	(2.39)	(0.92)	(1.29)	(3.80)	(0.55)	(2.41)	(1.10)	(1.27)	(0.53)	(1.02)	(0.51)	(0.64)
DS2 - 2.1 - CT	Run 1 (seed 11)	52.64	64.12	67.40	33.68	46.97	47.94	45.79	63.38	66.66	68.77	75.55	77.03	64.54	75.63	77.79
	Run 2 (seed 23)	51.77	66.46	67.65	33.96	48.06	47.72	47.96	63.71	65.76	69.03	76.45	76.97	68.8	76.05	76.66
	Run 3 (seed 47)	56.40	62.73	65.66	40.89	45.85	49.22	51.32	64.78	68.03	68.71	78.45	77.68	70.53	74.97	76.97
	Mean	53.60	64.44	66.90	36.18	46.96	48.29	48.36	63.96	66.82	68.84	76.82	77.23	67.96	75.55	77.14
	(Std.Dev)	(2.46)	(1.89)	(1.08)	(4.08)	(1.11)	(0.81)	(2.79)	(0.73)	(1.14)	(0.17)	(1.48)	(0.39)	(3.08)	(0.54)	(0.58)
DS2 - 2.1 - MD	Run 1 (seed 11)	55.34	65.66	67.75	37.55	47.66	48.97	48.02	60.58	62.43	69.16	76.19	78.52	68.77	75.74	75.29
	Run 2 (seed 23)	56.33	64.08	67.49	38.98	47.60	47.85	50.43	64.39	65.64	73.55	76.65	80.06	71.32	75.74	76.89
	Run 3 (seed 47)	57.33	69.42	66.17	38.14	48.00	48.19	52.13	64.69	65.29	72.90	78.45	78.45	69.51	75.18	76.89
	Mean	56.33	66.39	67.14	38.22	47.75	48.34	50.19	63.22	64.45	71.87	77.10	79.01	69.87	75.55	76.36
	(Std.Dev)	(1.00)	(2.74)	(0.85)	(0.72)	(0.22)	(0.57)	(2.07)	(2.29)	(1.76)	(2.37)	(1.19)	(0.91)	(1.31)	(0.32)	(0.92)
TransferQA - 2.1 - CT	Run 1 (seed 577)	48.94	60.87	65.34	31.93	38.95	41.35	49.75	59.84	62.82	70.77	74.52	75.74	68.95	72.58	75.95
	Run 2 (seed 17)	50.03	61.38	62.89	34.21	38.76	40.79	45.01	60.73	61.98	74.13	73.42	76.52	69.77	73.61	75.03
	Run 3 (seed 117)	51.77	60.51	64.60	31.24	39.36	43.82	46.59	56.92	61.92	68.45	75.48	75.94	68.32	73.32	75.39
	Mean	50.25	60.92	64.28	32.46	39.02	41.99	47.12	59.16	62.24	71.12	74.47	76.07	69.01	73.17	75.46
	(Std.Dev)	(1.43)	(0.44)	(1.26)	(1.55)	(0.31)	(1.61)	(2.41)	(1.99)	(0.50)	(2.86)	(1.03)	(0.41)	(0.73)	(0.53)	(0.46)

Table 15: Few-shot (1-5-10%) results on MultiWoZ 2.0 and 2.1 (*ver.*). CD, CT, MD each refer to *Cross-Domain*, *Cross-Task*, *Multi-Domain* few-shot scenarios. Full results and statistics of each run are provided here.

BART-Large *ver.* & `mode`		Attraction			Hotel			Restaurant			Taxi			Train		
		1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%	1%	5%	10%
DS2 - 2.1 - `CD`	Run 1 (seed 11)	53.15	62.51	63.79	33.99	45.51	49.22	46.95	59.66	63.32	68.58	76.52	79.55	56.68	73.69	74.89
	Run 2 (seed 23)	51.51	62.80	61.83	34.33	46.60	48.47	48.35	61.45	62.19	68.26	77.81	79.10	62.12	73.00	76.76
	Run 3 (seed 47)	55.50	65.59	60.16	34.80	46.22	47.94	50.58	61.66	64.45	69.23	76.84	80.84	63.09	73.13	76.74
	Mean (Std.Dev)	53.39 (2.01)	63.63 (1.70)	61.93 (1.82)	34.37 (0.41)	46.11 (0.55)	48.54 (0.64)	48.63 (1.83)	60.92 (1.10)	63.32 (1.13)	68.69 (0.49)	77.06 (0.67)	79.83 (0.90)	60.63 (3.46)	73.27 (0.37)	76.13 (1.07)
DS2 - 2.1 - `CT`	Run 1 (seed 11)	39.87	61.61	64.50	29.93	42.63	46.72	37.30	56.77	62.31	64.39	60.92	73.94	56.28	70.45	75.81
	Run 2 (seed 23)	39.20	61.70	59.74	32.49	44.07	46.16	39.77	59.90	59.81	61.74	63.32	76.06	64.17	69.58	74.00
	Run 3 (seed 47)	41.41	58.07	60.68	29.93	41.04	45.47	37.45	56.59	62.01	63.23	70.00	75.29	46.90	72.98	73.79
	Mean (Std.Dev)	40.16 (1.13)	60.46 (2.07)	61.64 (2.52)	30.78 (1.48)	42.58 (1.52)	46.12 (0.63)	38.17 (1.38)	57.75 (1.86)	61.38 (1.37)	63.12 (1.33)	71.27 (2.54)	75.10 (1.07)	55.78 (8.65)	71.00 (1.77)	74.53 (1.11)
DS2 - 2.1 - `MD`	Run 1 (seed 11)	42.06	61.32	58.14	30.49	38.92	45.13	38.58	51.83	61.51	61.16	65.74	66.65	54.31	72.95	68.35
	Run 2 (seed 23)	45.92	53.83	60.80	33.40	39.76	43.29	36.53	56.30	59.30	59.94	65.48	71.68	58.28	68.09	68.56
	Run 3 (seed 47)	41.03	55.40	56.66	32.62	41.92	47.75	39.60	62.10	53.32	60.77	65.23	67.81	60.91	68.85	72.29
	Mean (Std.Dev)	43.00 (2.58)	56.85 (3.95)	58.53 (2.10)	32.17 (1.51)	40.20 (1.55)	45.39 (2.24)	38.24 (1.56)	56.74 (5.15)	58.04 (4.24)	60.62 (0.62)	65.48 (0.26)	68.71 (2.63)	57.83 (3.32)	69.96 (2.61)	69.73 (2.22)

Table 16: Few-shot(1-5-10%) results on MultiWoZ 2.1 with BART-Large model. Meaning of the fields are same as in Table 15.

Few-shot ratio		**1%**	**5%**	**10%**
DS2 - T5 (2.0)	Run 1 (seed 11)	35.67	46.21	47.86
	Run 2 (seed 23)	38.22	46.01	47.79
	Run 3 (seed 47)	34.57	43.19	47.18
	Mean (Std. Dev)	36.15 (1.87)	45.14 (1.69)	47.61 (0.37)
DS2 - T5 (2.1)	Run 1 (seed 11)	32.04	43.30	44.30
	Run 2 (seed 23)	34.74	44.06	46.40
	Run 3 (seed 47)	34.50	45.24	45.43
	Mean (Std. Dev)	33.76 (1.49)	44.2 (0.98)	45.38 (1.05)
DS2 - BART (2.1)	Run 1 (seed 11)	27.52	37.39	40.05
	Run 2 (seed 23)	27.86	36.86	40.61
	Run 3 (seed 47)	29.37	38.88	40.21
	Mean (Std. Dev)	28.25 (0.98)	37.71 (1.05)	40.29 (0.29)

Table 17: Few-shot(1-5-10%) all-domain results on MultiWoZ 2.0 & 2.1 for multi-domain setting.