UNO-DST: Leveraging Unlabelled Data in Zero-Shot Dialogue State Tracking

Chuang Li[1,2], **Yan Zhang**[1], **Min-Yen Kan**[1], **Haizhou Li**[1,3]
[1]National University of Singapore
[2]NUS Graduate School for Integrative Sciences and Engineering
[3]Chinese University of Hong Kong, Shenzhen
lichuang@u.nus.edu
{eleyanz, kanmy, haizhou.li}@nus.edu.sg

Abstract

Previous zero-shot dialogue state tracking (DST) methods only apply transfer learning, ignoring unlabelled data in the target domain. We transform zero-shot DST into few-shot DST by utilising such unlabelled data via joint and self-training methods. Our method incorporates auxiliary tasks that generate slot types as inverse prompts for main tasks, creating slot values during joint training. Cycle consistency between these two tasks enables the generation and selection of quality samples in unknown target domains for subsequent fine-tuning. This approach also facilitates automatic label creation, thereby optimizing the training and fine-tuning of DST models. We demonstrate this method's effectiveness on general language models in zero-shot scenarios, improving average joint goal accuracy by 8% across all domains in MultiWOZ[1].

1 Introduction

Dialogue state tracking (DST) is a crucial task in understanding users' intentions by extracting the dialogue states from the dialogue history (Balaraman et al., 2021), where a single dialogue state is a pairing of a slot type (e.g.,<*hotel-name*>) and a slot value (e.g.,<*Hilton hotel*>), as in Figure 1. Dialogue states are a set of those combinations (e.g.,<*hotel-name: Hilton hotel*>) retrieved by DST models, given dialogue history and slot types. Traditional methods train and evaluate DST models with manually-labelled dialogue states in each domain, which can be costly and time-consuming (Wu et al., 2020b; Hosseini-Asl et al., 2020). Recently, DST under zero and few-shot settings draw increased attention (Lin et al., 2021b; Hudeček et al., 2021). Compared with few-shot methods, zero-shot approaches are more challenging, due to unseen slot types and data scarcity in unknown target domains.

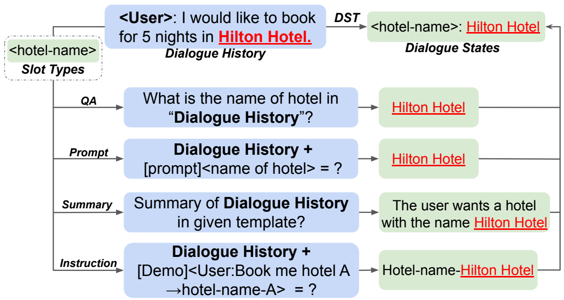


Figure 1: Examples of zero-shot methods in DST.

In both zero and few-shot settings, the majority of existing methods convert the DST problem into other common problem settings in natural language processing (NLP): for example, Question Answering (QA; Lin et al., 2021a; Li et al., 2021), prompt learning (Lee et al., 2021), summarization (Shin et al., 2022) and instruction learning (Gupta et al., 2022). For example in Figure 1, a given slot type (<*hotel-name*>) can be transformed into a QA setting by queries like "What is the hotel name mentioned in the dialogue history?" and the slot values can be predicted by a QA model accordingly.

Transfer learning methods also convert DST tasks to generation ones, more suited for pretrained language models (LMs; Devlin et al., 2019). However, such methods cannot fully leverage the capability of LMs in generation and selection. Two main difficulties emerge: 1) the performance of the chosen NLP tasks can be unpredictable for unseen slot types in a new domain due to domain divergence; and 2) existing models are only trained in the known domains, without utilizing any unlabeled data in the new target domain.

This work proposes **UNO-DST**[2], a method to leverage the **un**labelled data for **zero**-shot **DST** in target domain. Inspired by the popularity of multi-

[1]Code and data are available at https://github.com/lichuangnus/UNO-DST

[2]"Uno", Spanish for "one", embodies our proposed strategy in this paper: transitioning from zero to one and subsequently from one to all.

Findings of the Association for Computational Linguistics: NAACL 2024, pages 2972–2983
June 16-21, 2024 ©2024 Association for Computational Linguistics

task learning and self-supervised learning (Zhang and Yang, 2021; Tsai et al., 2021), UNO-DST employs a two-step training framework invoking both joint and self-training (Figure 3). Aside from the main task of generating slot values, we design an auxiliary task of generating slot types. We then jointly train both tasks using the labelled training data in source domains. For the self-training period, we implement the concept of cycle consistency within our two tasks (Zhu et al., 2017; Yang et al., 2023). That is, a text output from the main task serves as input to the auxiliary task, and the resultant text produced by the auxiliary task should match the original input text (Figure 2). This process forms a full cycle, ensuring consistent generation and selection of dialogue states from the unlabelled data, which is further used for fine-tuning the model. In this way, we convert zero-shot problems into few-shot ones. Importantly, our framework is model-agnostic which applies to different baseline models. Our main contributions are as follows:

- To the best of our knowledge, we are the first zero-shot DST work to use unlabelled training data in an unknown target domain;
- We introduce an auxiliary task to facilitate the training of the main task, the selection of fine-tuning samples, and the generation of unseen or new slot types;
- We demonstrate our methods with encoder-decoder LMs and large language models (LLMs), showing its effectiveness on two popular DST datasets.

2 Related Works

Existing DST methods are generally classified as either 1) full-data or 2) low-resource DST. Despite the method chosen, unlabeled training data in the target domain remains unexploited; in few-shot DST, although pseudo labels can be derived from unlabeled data in the same domain (Lee et al., 2023), there is a notable absence of research on target domain unlabelled data in zero-shot DST.

Full-data DST are commonly trained with fully annotated multi-domain conversations (Wu et al., 2020a; Hosseini-Asl et al., 2020). SOTA models focus on DST tasks with well-annotated datasets (Mrkšić et al., 2017; Ren et al., 2018). However, the annotation work for data in a new domain can be costly. Hence there is interest in transferring the knowledge of a model from a known domain into

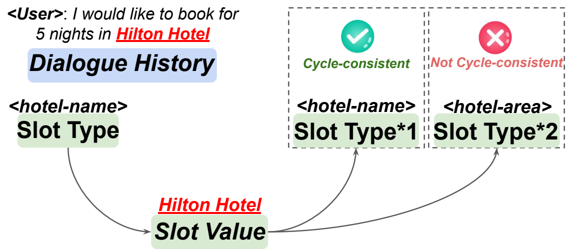


Figure 2: Cycle consistency in DST.

an unknown domain and conducting DST tasks in a low-resource setting.

Low-resource DST uses zero- or few-shot learning in the unknown target domain. Here, the state-of-the-art use a single NLP task to transfer knowledge from the source domains to the unknown target domain (Lin et al., 2021b; Shin et al., 2022). While transfer learning tasks achieve good results, each method is task-dependent. Thus, task-independent strategies have been proposed (Wang et al., 2022; Yang et al., 2023).

Multi-task Learning involves simultaneous training of a model on diverse tasks, to boost performance on trained downstream tasks. It also holds promise for enhancements on new tasks (Raffel et al., 2020; Zhang and Yang, 2021). However, existing methods typically neglect to assess the consistency across multiple tasks after joint training, while our approach leverages the cycle consistency for selection (Zhu et al., 2017; Wang et al., 2023).

3 Methodology

In Figure 3, we show an overview of **UNO-DST** with joint training and self-training periods. Our method includes two tasks: a main task for slot value prediction (§3.1) and an auxiliary task for slot type prediction (§3.2). In the joint training period, both tasks are jointly trained in the known source domains (§3.3). In the self-training period, we introduce three steps to generate dialogue states, select good samples, and fine-tune the LM (§3.4). Lastly, we elaborate on the transferability of our strategy with an oracular selection approach (§3.5).

3.1 Task Definition

The main task for DST is predicting the <*slot-type:slot-value*> pairs with given dialogue history and slot types from a pre-defined slot type list, as shown in Figure 1. For each domain, there are seen slot types which appear in other domains (e.g.,*"hotel-name" and "restaurant-name"*) or un-

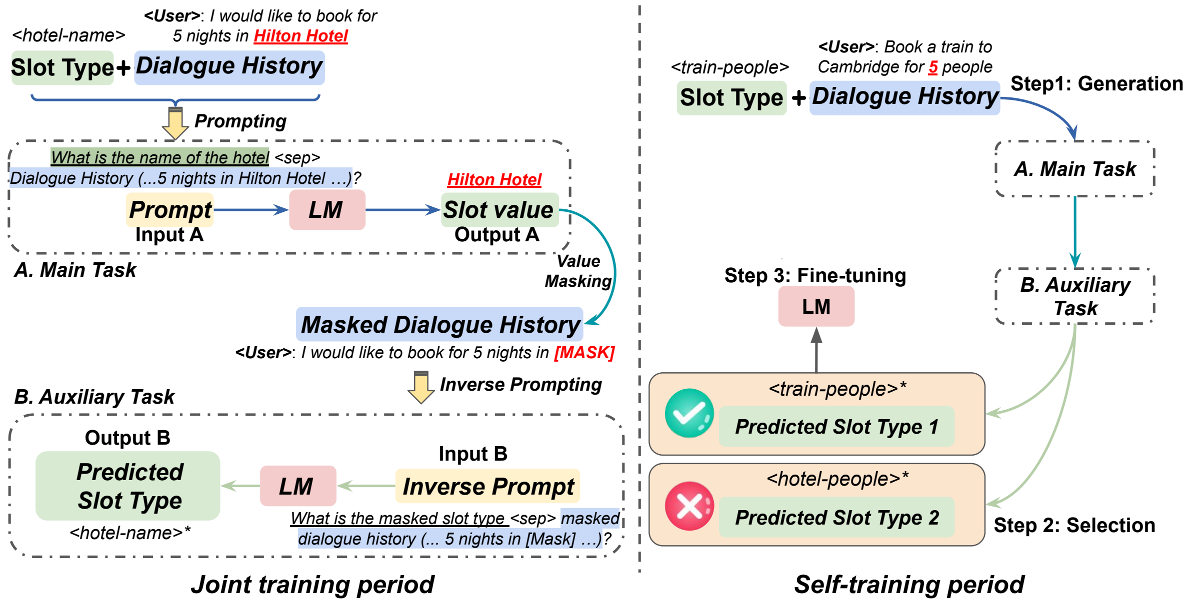


Figure 3: Overview of **UNO-DST** which consists of two periods: 1) joint training for both task A (slot value prediction) and B (slot type prediction), 2) self-training in the target domain. Step 1: Generation of slot values and types; Step 2: Selection of good samples with cycle consistency; Step 3: Fine-turning the LM with selected samples.

seen slot types which are unique in the specific domain (e.g.,*"hotel-stars"*). The number of these unseen slot types represents the difficulty of zero-shot DST for each domain (Wang et al., 2022).

We denote the dialogue history in a t-turn conversations as $C_t = \{c_1, c_2...c_t\}$ and slot types S in domain h as $S_h = \{s_1, s_2...s_n\}$. For each conversation turn, the main goal is to predict slot values v'. Therefore, the input for the LM is combined of dialogue history and slot types, with the output being slot values, as shown in Eq. 1.

$$v'_i = LM(C_t, s_i) \tag{1}$$

Compared with methods that select slot values from a constant ontology list using classification models (Shi et al., 2017), we enhance the capability of text-to-text LMs for text generation (Heck et al., 2020). For the case when there are no slot values related to a given slot type, we train the model to output a *"none"* value, indicating that there are no dialogue states from the current conversation turn.

To better utilise the capability of LMs in different tasks, we utilize different prompt functions "$P(.)$" to generate the prompt in the correct format. For example, given a slot s and context c, the QA prompt p for the DST can be p^{main} =*"What is the value of slot s in context c?"*. We formulate the way of using prompts for the DST main task as:

$$v'_i = LM(p_i^{main}) = LM(P(s_i, C_t)) \tag{2}$$

3.2 Auxiliary task

As joint training can improve the accuracy of LMs, we design an auxiliary task to facilitate the training of the main task (Zhang and Yang, 2021; Su et al., 2022; Yang et al., 2023). We propose an auxiliary task to help the model better understand the semantic and context information from the dialogue history in the joint training period and serve as a regulator to check the main task predictions obtained during the self-training period.

We design the auxiliary task as the inverse (converse) prompt of the main task. In opposition to the main task, the auxiliary task thus takes the slot values v as input and generates the slot types s' as outputs, which forms a cycle-consistent loop as a foil to the main task. To make it easier for LMs, we convert the slot values v and dialogue history C_t into a masked dialogue history C_t^m for the model to make better masked predictions, as in Eq. 3. The inverse QA prompt p^{aux} is generated as *"What is the masked slot type in context C_t^m?"* from inverse prompt function "$IP(.)$" (Figure 3). We implement the auxiliary task during both the joint training and self-training periods to facilitate slot values generation and selection.

$$s'_i = LM(p_i^{aux}) = LM(IP(v_i, C_t^m)) \tag{3}$$

3.3 Joint training with auxiliary tasks

We conduct a simple version of joint training with only two tasks: the main and the auxiliary DST

	MultiWOZ	SGD	
	JGA	JGA	AGA
Benchmarks	25.8	27.6	58.0
T5DST	32.4	NA	NA
SD-T5	35.6	NA	NA
TransferQA	35.8	21.3	60.8
UNO (JT)	**36.6** (+0.8)	**36.9** (+15)	**75.9** (+15)
UNO (JT-ST)	**40.8** (+5.0)	**47.4** (+26)	**81.8** (+21)

Table 1: Average zero-shot JGA and AGA results on MultiWOZ and SGD. JT/ST stands for joint/self-training and red figures calculate the performance increase of UNO-DST over TransferQA.

tasks. The training samples for the main tasks are created using dialogue history and slot type, while the samples for the auxiliary DST tasks are created by masking the slot values from dialogue history.

As the auxiliary task is an inverse process of the main task, the model is trained for the same knowledge in a cycle-consistent way. By predicting the masked slot type from the masked dialogue history, the model is familiar with the context and different slot types. With our specially designed auxiliary task, the generation model reuses the existing data for another round of training without the need to increase the amount of training data or model parameters. We formulate the loss function for the main task L_m and auxiliary task L_a as:

$$L_m = -\sum_{i}^{n} \log p(v_i'|C_t, s_i) \tag{4}$$

$$L_a = -\sum_{i}^{n} \log p(s_i'|C_t^m, v_i) \tag{5}$$

The final loss is a simple average of both. To keep the process simple, we do not add hyperparameters to the model framework. Importantly, as the auxiliary task samples are generated using the inverse prompt of the main task, the ratio of these two tasks mirrors the natural distribution of both tasks throughout the joint training period.

3.4 Self-training with auxiliary tasks

Compared with other zero-shot DST models, the key novelty of our strategy is in using the unlabelled training data in the unknown target domain for self-training. Self-training aims to generate pseudo labels and select data samples that further fine-tune the models. In the self-training period, we divide the strategy into three steps: termed generation, selection and fine-tuning.

Step 1 Generation tests both tasks using the unlabelled training data in the unknown target domain to generate predicted slot values v' and slot types s'. Auxiliary tasks in self-training are created by value masking, as shown in Figure 3. For training samples with slot values that do not directly copy from the original context (such as *"yes/no"* for *"hotel-parking"*), masking the slot value in the original context does not work. Such samples are omitted in creating the masked dialogue history.

Step 2 Selection tests the cycle consistency between main and auxiliary tasks by comparing the predicted slot types s' with the original slot types s in each dialogue turn. A simplified selection process is shown in Figure 3. In experiments, only the conversations with fully correct slot types are selected as good samples like joint goal accuracy settings, aiming to reduce the selection error.

Step 3 fine-tunes the model $LM(.)$ with selected samples and predicted slot values v'. This completes the conversion of zero-shot DST into few-shot DST, helping the model adapt to unknown domains without increasing data annotation and model parameters. For LLMs that are difficult to fine-tune, we propose other solutions (§ 7).

3.5 Oracular selection for zero-shot DST

Even though there are many studies working on zero-shot DST, to the best of our knowledge there are no common methods to identify the the peak performance that each model can potentially achieve (oracular performance). Here, we discuss our proposed algorithm with respect to the oracular selection, aiming to benchmark our method against oracular results for each model as an upper bound.

According to our self-training methods (§3.4), zero-shot DST can always be converted into a few-shot DST by selecting good samples with self-generated slot values for fine-tuning. Since cycle consistency cannot ensure 100% correct data selection, oracular performance comes when we select only the correct self-generated samples and use them for fine-tuning. We define such performance as the upper bound for the zero-shot DST model.

4 Experiments and Datasets

Dataset. We train and test our model on both MultiWOZ 2.1 (Budzianowski et al., 2018) and the Schema-Guided Dialogue (SGD; Rastogi et al., 2020). MultiWOZ and SGD have dialogues distributed in both training and testing distributions

Model	Checkpoint	Attraction	Hotel	Restaurant	Taxi	Train	Average
SD-T5	t5-small	33.9	19.9	20.8	66.3	37.0	35.6
TransferQA	t5-large	33.9	22.7	26.3	61.9	36.7	35.8
T5DST†	t5-small	30.5	19.4	20.4	**66.3**	25.6	32.4
UNO (JT)	t5-small	33.5	21.0	22.4	65.2	38.7	36.2
UNO (JT-ST)	t5-small	**36.1** (+5.6)	23.0	24.0	65.0	48.0	39.2
UNO (JT)	t5-QA	32.9	22.9	29.5	66.0	31.7	36.6
UNO (JT-ST)	t5-QA	33.1	**25.7** (+6.3)	**31.0** (+10.6)	65.5	**48.9** (+23.3)	**40.8** (+8.4)

Table 2: Zero-shot JGA results with different LM checkpoints. The lower/upper bound and best results for each domain are shown in bold. JT and ST stand for the results after joint- and self-training. † shows results of our replicated T5DST model, and red figures give the performance gap compared to †.

	TransferQA		UNO (JT)		UNO (JT-ST)	
Domains	**JGA**	**AGA**	**JGA**	**AGA**	**JGA**	**AGA**
Flights	03.6	42.9	**26.4**	**75.1**	25.3	72.7
RideSharing	31.2	61.7	33.3	64.3	**73.5**	**89.8**
Homes	**31.7**	**80.6**	16.8	77.6	17.9	76.3
Events	15.6	56.8	11.5	58.0	**23.1**	**71.6**
Movies	24.0	56.2	35.5	**86.7**	**52.6**	**86.7**
Services	37.2	75.6	75.1	92.1	**77.2**	**92.4**
Travel	14.0	24.2	55.2	76.7	**56.4**	**77.8**
Weather	40.3	59.4	93.8	98.0	**94.3**	**98.5**
Hotels	13.5	60.1	44.8	85.6	**75.9**	**94.6**
RentalCars	**10.8**	73.8	7.5	72.9	05.4	**79.4**
Restaurants	16.3	68.9	31.8	74.7	**35.9**	**78.5**
Media	30.2	67.5	37.0	69.7	**60.0**	**89.2**
Music	08.9	**62.4**	11.6	54.9	**19.1**	55.5
Average	21.3	60.8	36.9	75.9	**47.4**	**81.8**

Table 3: Zero-shot JGA and AGA results for domains in SGD dataset. **Bold** shows the best results and JT/ST stands for joint/self-training.

over 7, 13 domains, representing 7K, 16K training examples in English, respectively. We use standard means for data pre-processing (Budzianowski et al., 2018) and follow the MultiWOZ leave-one-out settings for zero-shot training and testing in both datasets (Wu et al., 2019; Rastogi et al., 2020).

Evaluation metrics. The primary metric for DST evaluation is joint goal accuracy (JGA), which compares the set of generated predicted values with the set of ground truth ones after each conversation turn and average goal accuracy (AGA) calculates the JGA only for active slot types as in SGD dataset (Henderson et al., 2014; Rastogi et al., 2020).

Baselines and experiment setup. We use T5 (Raffel et al., 2020) as our baseline model. For a fair analysis, we also compare our results with previous DST benchmarks: TRADE (Wu et al., 2019) and the SGD baseline (Rastogi et al., 2020), and current SOTA models: T5DST (Lin et al., 2021b), TransferQA (Lin et al., 2021a) and SD-T5 (Wang et al., 2022). We adopt the cross-domain settings (Wu et al., 2019) for both datasets, experimenting on two checkpoints, "t5-small"[3] and "t5-QA"[4]. The unlabelled training data in the target domain will be used for self-training and testing data in target domain is only used for final testing.

We select QA as the main task in our framework for its popularity and test it through different checkpoints holding parameters fixed. We adopt the open-source "t5-small" (Raffel et al., 2020) with 60M parameters as our baseline and train using AdamW with a learning rate of 0.0001, batch size 8 for 1 epoch (zero-shot setting) and 3 epochs (fine-tuning setting) on single GeForce RTX3090.

5 Results

Table 1 analyzes the zero-shot results of our UNO-DST and the baselines, including the state-of-the-art (SOTA) TransferQA model, across both datasets. Our model surpasses all baselines for both joint and self-training phases, inclusive of TransferQA using "t5-large". Detailed outcomes for each domain and specific training periods across each dataset are presented in Tables 2 and 3.

5.1 Joint training results

For the joint training period in the MultiWOZ dataset (Table 2), we are using the same model and prompt as T5DST. UNO-DST shows an increase of more than 4% for JGA across all checkpoints. For SGD (Table 3), our joint training period increases the previous baseline by even larger margins of 15.6% in JGA (15.1% in AGA). The joint training period is critical as it prepares a model for self-training. Table 2 shows that using different model checkpoints is also critical even when using the

[3] https://huggingface.co/t5-small
[4] https://github.com/facebookresearch/Zero-Shot-DST

Round	Att.	Hotel	Res.	Taxi	Train	Avg	Gain ↑
0	32.86	22.91	29.47	66.00	31.68	36.58	
1	33.09	25.66	30.99	65.48	48.90	40.82	(+4.24)
2	35.53	27.22	31.44	64.71	54.60	42.70	(+1.88)
3	36.62	27.09	31.14	65.48	53.31	42.73	(+0.03)

Table 4: JGA for multiple rounds of self-training on MultiWOZ. Absolute gains indicated in red.

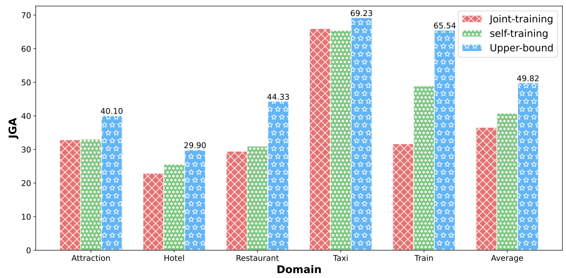


Figure 4: Gains by joint and self-training stages of UNO-DST on the "t5-QA" checkpoint. We show the results of oracular selection (Upper-bound) in each domain for relative comparison.

same model architecture and parameter size. The model performs best when we follow the prompt format in each baseline.

5.2 Self-training results

For the MultiWOZ dataset (Table 2), self-training further improves average JGA by 3.09% after the joint training period. Compared with the baseline, the best performance increases by 8.38% in JGA. In SGD dataset (Table 3), self-training improves the average JGA and AGA in 12 out of 13 domains by an average of 10.5% and 5.9% compared with the joint-training alone, and over 26% and 21% compared to the baseline.

The success of self-training proves the possibility of using pseudo labels generated from zero-shot DST models to bootstrap performance. However, carefully selecting good samples to fine-tune the model is challenging because not all the domains benefit from the self-training process. For example, the result for the *"Taxi"* domain in MultiWOZ and *"Flights"* domain in SGD decreased after self-training. We examine the rationale behind the gains obtained through self-training, which is associated with the gap between joint training and oracular results, as further discussed in § 6.

As shown in Table 4, as we lengthen self-training from a single round to multiple rounds, our framework's performance continues to improve. However, the performance gap between results from different rounds shows diminishing returns, signalling a plateau. The best result with UNO-DST comes when adding more variation is insignificant and so we stop the training when the margin is below 0.1 in JGA. Future work is required to systematically study this strategy over multi-round self-training.

6 Discussion

6.1 Oracular selection

For the oracular calculation, we select only the 100% correct samples from the zero-shot predictions and use them for fine-tuning. In Figure 4, we visualise the gains of joint training and self-training alongside our upper bound. While efficacy differs from domain to domain, an important observation is that when the margin between the upper bound (blue columns) and joint training (red columns) is large, the model has a larger gain from self-training, as in *"Train"* domain. In contrast, for *"Taxi"* domain, the influence of self-training is weak (cf § 5.2 Self-training results). Utilizing upper bounds calculations enables us to swiftly evaluate whether a domain or model is apt for the self-training period. In other words, a larger margin between joint training and the upper bound yields a larger potential improvement that the model can achieve with fine-tuning or self-training strategy.

6.2 Unseen slot type prediction

For each domain, there are seen slot types which appear in other domains (e.g.,*"hotel-name" and "restaurant-name"*) or unseen slot types which are unique in the specific domain (e.g.,*"hotel-stars"*). The ratio of the occurrences of these slot types represents the difficulty of zero-shot DST for each domain (Wang et al., 2022). As shown in table 5, the original setting for the MultiWoz dataset has 30 given slot types. However, not all of them appear in every domain. For some certain domains, like the "hotel" domain, there are 4 unique slot types which do not appear in other domains, including "stars", "internet", "stay" and "parking". In Figure 5, we show the slot accuracy for the hotel domain. It shows that generally, the unseen slot types will perform worse than the seen slot types (Wang et al., 2022). Prediction for those slot types in zero-shot cross-domain settings can be challenging as there is no further information from the other source domains. In addition, half of the unseen slot types in the "hotel" domain are related to "yes/no" slot val-

All Given Slot Types in MultiWOZ 2.1
$area^{123}, arriveby^{45}, day^{235}, departure^{45}, destination^{45}, food^{3}, internet^{2}, leave^{45}, name^{123}, people^{235}, parking^{2}, price^{23}, stars^{2}, stay^{2}, time^{3}, type^{12}$
Seen Slot Types in Hotel Domain
$area^{123}, day^{235}, name^{123}, people^{235}, price^{23}, type^{12}$
Unseen Slot Types in Hotel Domain
$internet^{2}, parking^{2}, stars^{2}, stay^{2}$

Table 5: Seen and unseen slot types in hotel domain. The superscript on each slot type indicates the domain information from: (1:attraction, 2:hotel, 3:restaurant, 4:taxi, 5:train)

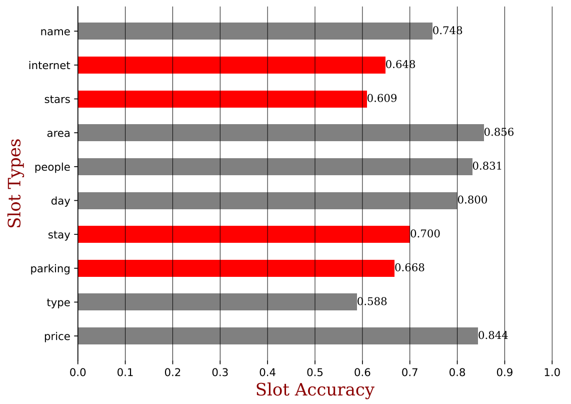


Figure 5: Slot Accuracy for (grey) seen and (red) unseen slot types in the hotel domain.

ues, whereas in our joint training settings in § 3.3, we skip the masking of those "yes/no" values from the context and the model is less trained compared with other slot types. We hope that future works will improve on "yes" or "no" value prediction.

6.3 New slot type generation

All the existing zero-shot DST methods require given slot types in generating the slot values for both source and target domains and our model also follows the same experiment settings (cf § 3.1). However, our model can also self-generate reasonable slot types either in or beyond the 30 given slot types with our designed auxiliary task. To self-generate new slot types, we do a case study on the MultiWoZ *"train"* domain and perform random word masking for all the dialogue history, inputting those randomly masked dialogue histories to the auxiliary task for slot type predictions, as shown in Figure 6b. In Table 6, we show some valid new slot types generated by our auxiliary tasks with dialogue history. For example, *"asking for the ticket price"* in *"train"* domain and *"asking for parking information"* in *"Restaurant"* domain are reasonable new slot types, which can also be included.

6.4 DST without pre-defined slot types

As discussed in the previous section, the auxiliary task can facilitate the generation of new slot types beyond the pre-defined ones in MultiWOZ. Besides adding more slot types to the given slot type list, we believe that our proposed model can conduct DST tasks in an unknown target domain without any given slot types and we describe the proposal of zero-shot DST without slot types in Figure 6.

In order to eliminate the use of pre-defined slot types, we add a slot type generation period between joint and self-training, which identifies and select domain-relevant slot type corpus. Similar to the process proposed by Hudeček et al. (2021), we can first use our auxiliary task to generate potential slot types based on random masked dialogue history, as shown in Figure 6b. The generated text may contain domain-irrelevant or similar slot types and we propose a weak selection and merging of task-relevant and similar slot types for slot type corpus (Hudeček et al., 2021). Secondly, those generated slot-type corpus can be used for self-training in the unknown target domain, as discussed in § 3.4.

During our testing, our auxiliary task can generate predictions including all 6 given slot types in the *"train"* domain (Table 6.3), as well as valid slot types in other domains, which demonstrates the potential of future zero-shot methods without pre-defined slot types. We look forward to future works for zero-shot DST without any labelled data in slot types and values in unknown target domains.

7 UNO-DST with ChatGPT

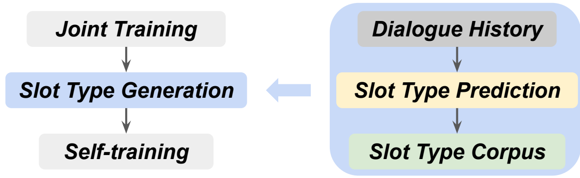
While earlier sections analyse the effectiveness of our methods on LMs like "T5", this section focuses on the potential application of UNO-DST with large language models (LLMs), such as Brown et al., 2020 and Touvron et al., 2023. Specifically, we examine UNO-DST for zero-shot DST using OpenAI's ChatGPT[5] as the backbone LLM, an LLM which has been adopted as a language tool for information extraction with strong capabilities, even without specific training or fine-tuning.

Our study will test the efficacy of our self-training strategy in **UNO-DST** on ChatGPT, includ-

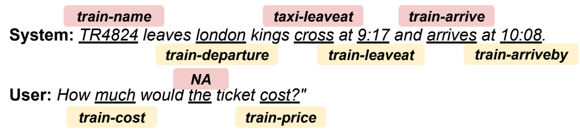
[5]https://chatgpt.openai.com

All Generated Slot Types in Train Domain
$people^5, day^5, destination^5, departure^5, leave^5, arrive^5, price^5, type^5, time^5, area^5, name^5$
Valid New Slot Types
$price^{15}, day^1, parking^3, name^5$
Dialogue Example [PMUL1359] $(price^5, name^5)$
System: *"Okay, tr6572 departs at 05:29."* **User:** *"What is the price?"*
Dialogue Example [PMUL3027] $(parking^3)$
System: *"I have 2 Turkish restaurants in the centre?"* **User:** *"Do they offer free parking?"*
Dialogue Example [PMUL1118] (day^1)
User: *"I am in Cambridge for the week and want to know what museums you guys have there."*

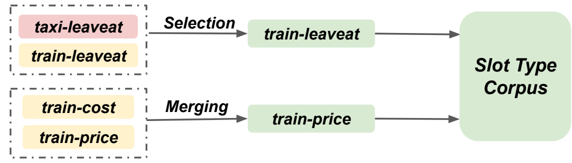
Table 6: Newly-generated slot types with examples. The superscript on each slot type indicates the domain information from: (1:attraction, 2:hotel, 3:restaurant, 4:taxi, 5:train)



(a) Overall system design for zero-shot DST w/o slot types



(b) Slot type prediction with randomly masked tokens



(c) Building slot type corpus with merging and selection

Figure 6: Zero-shot DST without pre-defined slot types

ing the web interface for conversational approach (§7.1) and the API[6] for larger testing corpus using in-context learning (ICL; Hu et al., 2022; §7.2).

7.1 Conversational approaches

Implementation. We skip the joint training for ChatGPT and use conversations as an inference approach. As shown in Figure 7, we implement main and auxiliary tasks with conversations asking for slot values or types. The selection and fine-tuning steps in the self-training strategy have been converted into the correction or confirmation step, where we provide the slot value and type predictions from the previous two questions to ChatGPT and ask whether it needs to revise the slot value to the main task. We consider the revised response from ChatGPT as the final answer to our main task. We manually examine all the responses generated and give examples of their performance.

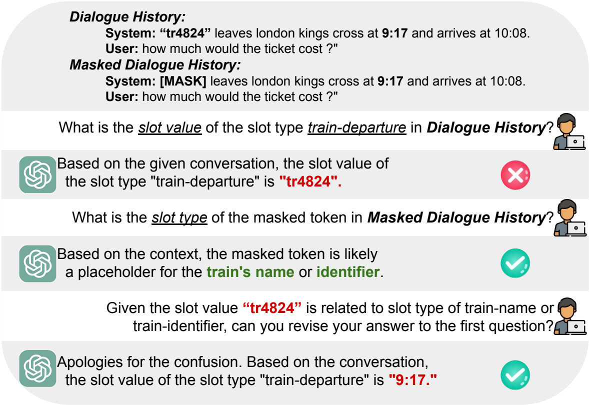
Results and discussion. We show two cases of predictions made by ChatGPT using the same dialogue history in Figure 7a and 7b. In Figure 7a, ChatGPT first made a wrong main task prediction, followed by a correct but not consistent auxiliary task prediction. When we provide all historical information to ChatGPT and ask for a revised main task prediction, it realised that the slot value from the first prediction is not consistent with the original slot type and it self-corrected its wrong prediction. In Figure 7b, we show another case of correct predictions which happens for the majority of the conversations. When ChatGPT is able to make correct predictions for both the main and auxiliary tasks, it confirms the correct predictions for the final question based on cycle consistency.

We illustrate how the cycle consistency between the main and auxiliary tasks aids ChatGPT in rectifying incorrect answers or confirming correct answers (Zhu et al., 2017). The strategy applies to the free accessible web interface of ChatGPT and is easy to implement. However, as conversations are difficult to quantify and evaluate, we only qualitatively show the results and encourage future research to explore all the potential implementations of cycle consistency in LLMs (Wang et al., 2023).

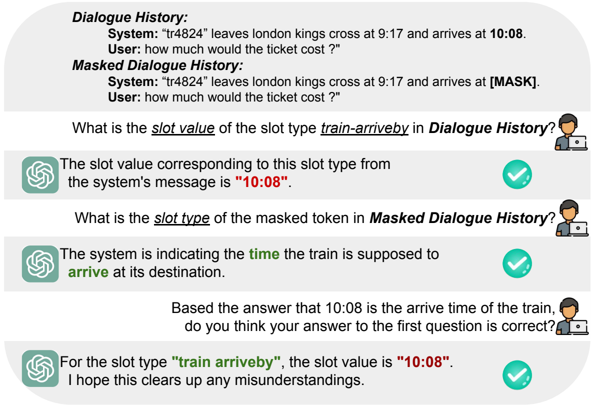
7.2 In-context Learning

Implementation. Following the settings in § 7.1, we skip the joint training period and apply our strategy to improve ChatGPT by ICL, providing ICL instructions and ICL examples in the ICL prompt. Specifically, we compare the JGA results between two different ICL prompts using the same instructions but different ICL examples: 1) ICL examples are composed of both dialogue history and dialogue states in the source domains because no dialogue states in the target domain are available for zero-shot DST settings and 2) ICL examples with dialogue history in the target domain and dialogue states generated using the conversational

[6]ChatGPT API model: gpt-3.5-turbo-0301



(a) Correction of the wrong prediction for "train-departure".



(b) Confirmation of the correct prediction for "train-arriveby".

Figure 7: Case studies for conversations with ChatGPT for zero-shot DST. (Red: Slot Values; Green: Slot Types)

approach in § 7.1. To illustrate, without cycle consistency, the ICL prompt is originally built by DST examples from the source domains and it is hard for the LLM to test on the target domain. After selecting good samples with the strategy discussed in § 7.1, the ICL prompt can be updated with examples in the target domain. We conduct small-scale experiments 3 times with the *"train"* domain in the MultiWOZ dataset by randomly sampling 100 conversations and inference with ICL prompts using different ICL prompts, evaluated by JGA.

Results and discussion. The resulting average JGA for the original source domain ICL prompt is 34.92% while the JGA for the selected target domain ICL prompt is 54.18%. Our self-training strategy works very well, serving the LLM to select valuable in-domain examples for the ICL prompt, improving the zero-shot DST performance by a large 19.25% margin. By manually examining the generated dialogue states, the ICL prompt modified with our strategy performs better, especially for conversations with longer dialogue turns and more slot types. Our self-training strategy demonstrates its capability in generating and selecting dialogue state samples in LLMs which can further improve the performance of zero-shot DST using an ICL prompt. However, due to the scope of this paper, we only test the application of our **UNO-DST** on ChatGPT with a small data corpus.

In summary, this section extends the applicability of the **UNO-DST** strategy to LLMs, assessing its efficacy in both conversational approaches and ICL. Conversational methods offer a straightforward mechanism for rectifying in-discussion errors, whereas ICL, leveraging APIs or LLM inferences, facilitates handling larger data corpora. Besides testing the cycle consistency strategy in well-suited DST tasks, additional work is required to extend it to other LLMs or more general NLP problems.

8 Conclusion

We propose a novel approach to convert the zero-shot DST into a few-shot setting by generating and selecting quality dialogue states from unlabeled data in the target domain through joint and self-training periods. We introduce and demonstrate how our proposed auxiliary task, which generates slot types as the inverse prompt for the main task which generates slot values, serves the whole model for 1) better accuracy of the main task in joint training 2) quality data selection in the self-training period 3) new slot types generation beyond the given slot type list and 4) upgrading to LLMs.

Our proposed strategies of **UNO-DST** are task-independent, which can be extended to other prompt formats and generalised to LLMs. We look forward to future works that engage additional auxiliary tasks which target new datasets and apply zero-shot DST, even where no slot types are given.

Limitations

This work has 3 limitations: 1) due to the limitation of computational resources, we only conduct experiments on small encoder-decoder LMs, which is "t5-small" and simple NLP tasks, which is "QA". Our future works will include more NLP tasks with different LMs to systematically test the performance of our proposed models. 2) Our reported self-training results are only for a single round of self-training because we could not find a way to continuously increase the performance of self-training. Our future plan seeks to improve and examine the best criteria for self-training using an early-stopping strategy. 3) The experimental settings for ChatGPT can be improved in three aspects: a) a larger data corpus can be applied with better instruction prompts in order to limit ChatGPT in generating more accurate values, b) an open-source LLM (Touvron et al., 2023) can be applied to better evaluate and replicate the results of the experiment, and c) a full self-training strategy including generation, selection, and fine-tuning can be tested to demonstrate the best performance with LLMs.

Ethical Concerns

Our self- and joint-training tunes models to amplify signals from the original dataset. While this strategy does work well in our experiments, if the dataset's signal is weak to start with, our methods may incorrectly amplify errors or biases. The application of our techniques in practical settings should be evaluated before deployment. This work experimented with publicly available datasets which require no additional annotation from humans.

Acknowledgement

The author thanks the anonymous reviewers for their valuable advice and Taha Aksu for his diligent editing of the manuscript.

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. Association for Computational Linguistics.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, 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*, pages 5016–5026, Brussels, Belgium. Association for Computational Linguistics.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Raghav Gupta, Harrison Lee, Jeffrey Zhao, Yuan Cao, Abhinav Rastogi, and Yonghui Wu. 2022. Show, don't tell: Demonstrations outperform descriptions for schema-guided task-oriented dialogue. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4541–4549, Seattle, United States. Association for Computational Linguistics.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. Trippy: A triple copy strategy for value independent neural dialog state tracking. *arXiv preprint arXiv:2005.02877*.

Matthew Henderson, Blaise Thomson, and Jason D. Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 263–272, Philadelphia, PA, U.S.A. Association for Computational Linguistics.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *CoRR*, abs/2005.00796.

Yushi Hu, Chia-Hsuan Lee, Tianbao Xie, Tao Yu, Noah A. Smith, and Mari Ostendorf. 2022. In-context learning for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 2627–2643, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Vojtěch Hudeček, Ondřej Dušek, and Zhou Yu. 2021. Discovering dialogue slots with weak supervision. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 2430–2442. Association for Computational Linguistics.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural*

Language Processing, pages 4937–4949, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Jihyun Lee, Chaebin Lee, Yunsu Kim, and Gary Geunbae Lee. 2023. Tracking Must Go On : Dialogue State Tracking with Verified Self-Training. In *Proc. INTERSPEECH 2023*, pages 4678–4682.

Shuyang Li, Jin Cao, Mukund Sridhar, Henghui Zhu, Shang-Wen Li, Wael Hamza, and Julian McAuley. 2021. Zero-shot generalization in dialog state tracking through generative question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 1063–1074, Online. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Zhenpeng Zhou, Paul Crook, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, and Pascale Fung. 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, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Seungwhan Moon, Paul 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, Online. Association for Computational Linguistics.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1777–1788, Vancouver, Canada. Association for Computational Linguistics.

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. *J. Mach. Learn. Res.*, 21(1).

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8689–8696.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2780–2786, Brussels, Belgium. Association for Computational Linguistics.

Hongjie Shi, Takashi Ushio, Mitsuru Endo, Katsuyoshi Yamagami, and Noriaki Horii. 2017. Convolutional neural networks for multi-topic dialog state tracking. *Dialogues with Social Robots: Enablements, Analyses, and Evaluation*, pages 451–463.

Jamin Shin, Hangyeol Yu, Hyeongdon Moon, Andrea Madotto, and Juneyoung Park. 2022. Dialogue summaries as dialogue states (DS2), template-guided summarization for few-shot dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3824–3846, Dublin, Ireland. Association for Computational Linguistics.

Yixuan Su, Lei Shu, Elman Mansimov, Arshit Gupta, Deng Cai, Yi-An Lai, and Yi Zhang. 2022. Multi-task pre-training for plug-and-play task-oriented dialogue system. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4661–4676, Dublin, Ireland. Association for Computational Linguistics.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Yao-Hung Hubert Tsai, Yue Wu, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2021. Self-supervised learning from a multi-view perspective.

Qingyue Wang, Yanan Cao, Piji Li, Yanhe Fu, Zheng Lin, and Li Guo. 2022. Slot dependency modeling for zero-shot cross-domain dialogue state tracking. In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 510–520, Gyeongju, Republic of Korea. International Committee on Computational Linguistics.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*.

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

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

Yuting Yang, Wenqiang Lei, Pei Huang, Juan Cao, Jintao Li, and Tat-Seng Chua. 2023. A dual prompt learning framework for few-shot dialogue state tracking. In *Proceedings of the ACM Web Conference 2023*, WWW '23, page 1468–1477, New York, NY, USA. Association for Computing Machinery.

Yu Zhang and Qiang Yang. 2021. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609.

Jun-Yan Zhu, Taesung Park, Phillip Isola, and Alexei A Efros. 2017. Unpaired image-to-image translation using cycle-consistent adversarial networks. In *Computer Vision (ICCV), 2017 IEEE International Conference on*.