FFS: Few-Shot Language Feedback for Domain Adaptation in End-to-End Dialogue State Tracking

Anonymous ACL submission

Abstract

End-to-end task-oriented dialogue (TOD) systems have become increasingly feasible due to advancements in language modeling. However, tasks such as dialogue state tracking (DST) remain challenging, particularly in domain adaptation, where models must generalize to unseen domains without additional supervision. While Large Language Models (LLMs) exhibit strong fine-tuning performance and even generalization, they still make mistakes and it can be difficult to correct those errors through fine-tuning. In this work, we propose a method that enables improvement of a fine-tuned LLM by incorporating few-shot language feedback. Our approach follows a two-step process: first, we bootstrap a draft model using data augmentation techniques to improve schema robustness. This model is then applied to a validation set, where incorrect predictions are identified. In the second step, expert annotators provide targeted natural language feedback on a subset of these errors, explicitly guiding the model on how to improve its performance on the task. The model is then fine-tuned again on both the original data and the feedback-augmented examples. Experiments on MultiWOZ and SpokenWOZ demonstrate that integrating language feedback in this manner improves DST performance by up to 5.8% in unseen domains.

1 Introduction

Dialogue State Tracking (DST) plays a pivotal role in task-oriented dialogue (TOD) systems by maintaining a structured representation of the user's goals, intents, and preferences as a conversation progresses. As these systems interface with external APIs, such as booking platforms or food ordering services, accurate DST is crucial for ensuring successful goal-oriented interactions. Traditional DST models were domain-specific, requiring large amounts of annotated data for every new domain. Recent work has addressed this limitation by leveraging multi-task instruction-tuned large language

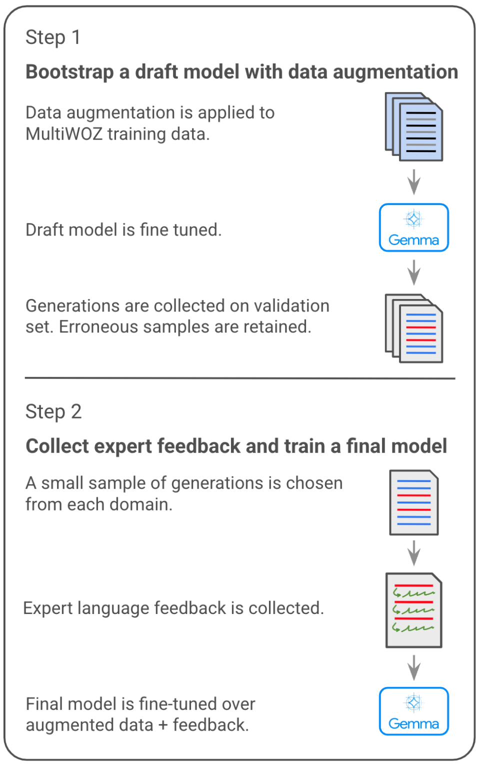
Step 1
Bootstrap a draft model with data augmentation
Data augmentation is applied to MultiWOZ training data.
Draft model is fine tuned.
Gemma
Generations are collected on validation set. Erroneous samples are retained.
Step 2
Collect expert feedback and train a final model
A small sample of generations is chosen from each domain.
Expert language feedback is collected.
Final model is fine-tuned over augmented data + feedback.
Gemma

Figure 1: Overview of our few-shot feedback method.

models (LLMs) for zero-shot DST. Zero-shot DST models track dialogue states for unseen domains without any fine-tuning on data samples from those domains (Aksu et al., 2023; Li et al., 2021; Feng et al., 2023; Yi et al., 2024; Hosseini-Asl et al., 2020; Heck et al., 2024). However, despite the advances in language modeling capabilities, recent work has provided evidence that zero-shot DST remains an open problem (Heck et al., 2023). To address this challenge, we propose **FFS** (Feedback in Few-Shot), a novel fine-tuning approach utilizing few-shot language feedback to enhance zero-shot domain adaptation for end-to-end DST.

Our contributions are as follows:

1. We propose Feedback in Few-Shot (FFS) for learning from few-shot language feedback

to improve zero-shot domain adaptation for DST.

2. We demonstrate that FFS provides an improvement in the Goal Accuracy by up to 16% on MultiWOZ and up to 6.7% on SpokenWOZ relative to the prior state-of-the-art.

3. We perform a careful ablation study revealing the source of improvements in FFS

Our findings highlight the potential of natural language feedback as a powerful supervision signal for improving generalization in LLM-based DST models. By leveraging small-scale expert corrections, we demonstrate that models can achieve significant improvements without requiring additional domain-specific labeled data.

2 Related Work

The use of language feedback is motivated by prior work that has demonstrated the ability of language models to refine their outputs given feedback critiquing their generations. Existing research has focused on integrating feedback from either language models as in SELFREFINE (Madaan et al., 2024), DCR (Wadhwa et al., 2024), LLM-AUGMENTER (Peng et al., 2023), McCallum et al. (2023) and Shepherd (Wang et al., 2023) or by integrating feedback from humans in-the-loop (Weston, 2016; Fidler et al., 2017; Scheurer et al., 2022; Li et al., 2022; Richardson et al., 2023).

While prior work has explored learning from language feedback, the concept of *few-shot language feedback* remains underexplored. Much of the prior methods that have been developed require crowdsourcing or AI generation for acquiring feedback, which is unsuitable in applications where detailed, expert knowledge is needed. In some situations, acquiring quality feedback can be very costly, e.g. when detailed knowledge of a niche dataset or task is needed, and only a few individuals are familiar enough to provide that feedback. To the best of our knowledge, no prior work has investigating the application of true few-shot feedback, i.e. only a handful (on the order of 10) of feedback samples are available. Our research aims to fill this gap by investigating the efficacy of few-shot language feedback in enhancing zero-shot domain adaptation for DST. By integrating minimal expert feedback into the training process, we seek to improve model generalization to unseen domains without the need for extensive domain-specific data.

3 Method

The current state-of-the-art in zero-shot domain adaptation for DST is *Encoding Schema Augmentation* (ESA) (Richardson et al., 2024), a method for improving generalization by augmenting the dataset with an encoded schema to encourage the model to pay attention to slot descriptions and values. We build off ESA for our bootstrapping, and then extend this idea to the few-shot feedback case, using augmentation to boost the significance of the feedback at training time.

Once we have bootstrapped a model using fine-tuning and ESA, we collect language feedback that directly targets mistakes made by the model in order to further improve performance. This process involves selecting a curated set of erroneous outputs, ranking them for informativeness, and then annotating them with detailed natural language corrections.

Feedback Collection The dataset $\mathcal{X}$ consists of pairs of dialogues x and dialogue states y. There exists a target domain T, and a corresponding subset of our dataset $\mathcal{X}_T$. Given an LLM π, we define the set of incorrect predictions over a dataset $\mathcal{X}$ as:

$$\mathcal{E} = \{(x, y) \in \mathcal{X} : \pi(x) \neq y\}. \tag{1}$$

To ensure domain equity in the feedback, we divide $\mathcal{E}$ into subsets for each d domain, $\mathcal{E}_d \subset \mathcal{E}$ containing errors in that domain. Each author independently selected a subset of $\mathcal{E}_d$, ranking instances based on the potential informativeness of feedback (at the author's discretion). Let S_1 and S_2 be the ranked sets chosen by the first and second author, respectively. The final set of feedback samples for domain d, $\mathcal{F}_d$, was determined by selecting the top k samples from the intersection of ranked choices:

$$\mathcal{F}_d = \text{Top-}k(S_1 \cap S_2). \tag{2}$$

For our experiments, we use $k = 5$ to simulate the scenario where only a handful of samples are available.

Feedback Annotation For each $(x, y) \in \mathcal{F}_d$, two different authors provided independent feedback sentences $f_1(x, y)$ and $f_2(x, y)$, describing the model's error and the correct reasoning. A third co-author then reviewed both annotations and selected the preferred feedback:

$$f^*(x, y) = \text{Select}(f_1(x, y), f_2(x, y)). \tag{3}$$

Thus, our final annotated feedback set is:

$$\mathcal{F}_d^* = \{(x, y, f^*(x, y)) \mid (x, y) \in \mathcal{F}_d\}. \tag{4}$$

Dialogue	Dialogue State	Model Response	Feedback

USER: Hello I'd like to book a table for Friday.
SYSTEM: I see openings at Chotchkie's all evening, on the west side. Is that ok?
USER: Yes, let's do 19:00.
SYSTEM: Great, it's booked. Can I help you with anything else?
USER: I also need a place to stay near there for that night.
	`restaurant-day: friday`
	`restaurant-time: 19:00`
	`restaurant-area: west`
	`restaurant-name: Chotchkie's`
	`hotel-day: friday`
	`hotel-area: west`
		`restaurant-day: friday`
		`restaurant-time: 19:00`
		`restaurant-area: west`
		`restaurant-name: Chotchkie's`
		`hotel-day: friday`
		`hotel-area: -`
			You can infer the hotel area because the user said they want it near the restaurant, which we know is in the west.

Table 1: Example of expert feedback written for a dialogue with an erroneous model response.

Integration into Training The collected feedback is incorporated into training as additional few-shot exemplars. We experiment with two methods of choosing the domains to include: **gFFS** (global-FFS), where examples are chosen from $\mathcal{F}_d$ for every d; and **pFFS** (pertinent-FFS), in which only domains present in the dialogue are included. We denote the union of the chosen $\mathcal{F}_d^*$'s as $\mathcal{F}^*$. For an input query x, the model conditions on $\mathcal{F}^*$ to generate an updated prediction:

$$\pi^*(x) = \arg\max_y P(y \mid x, \mathcal{F}^*). \tag{5}$$

This is the objective used to train the draft model until convergence, resulting in the final model (Figure 1.

4 Experiments

4.1 Metrics

Joint Goal Accuracy (JGA) Standard in the DST literature (Henderson et al., 2014), JGA measures the fraction of dialogue turns in which the entire dialogue state is correctly predicted.

Target Goal Accuracy (TGA) TGA (Richardson et al., 2024) measures the fraction of turns with nonempty target domain slots in which **all** target domain slots are accurately predicted.

4.2 Baselines

We compare our approach against several baselines for zero-shot domain adaptation in DST. **TransferQA** (Lin et al., 2021a) formulates DST as a question-answering (QA) problem, where the model extracts slot values based on schema-driven natural language questions. **T5DST** (Lin et al., 2021b) treats the task as sequence-to-sequence generation to predict slot values conditioned on the dialogue and schema. **D3ST** (Zhao et al., 2022) enhances zero-shot generalization by utilizing slot and value descriptions in the prompt, allowing the model to dynamically adapt to new domains without additional fine-tuning. **Prompter** (Aksu et al., 2023) and **DualLoRA** (Luo et al., 2024) introduce parameter-efficient approaches to understand dialogue and adapt to the schema. **Encoding Schema Augmentation** (Richardson et al., 2024) (ESA) introduces a data augmentation approach that encourages a stronger attention to slot descriptions and values in the schema rather than the slot names.

4.3 Modeling

For all our experiments we use `gemma-2-9b-it` (Gemma Team, 2024). Our prompt includes both the dialogue text as well as the schema (slot names, descriptions, and possible values) as additional contextual information. An example is provided in Appendix A.1. Our outputs are fomatted as JSON. Hyperparameters and compute details are included in Appendix A.

4.4 Datasets

We conduct our experiments on two open-source dialogue state tracking datasets. **MultiWOZ** (Budzianowski et al., 2018) is a multi-domain task-oriented dialogue dataset comprising annotated dialogues across eight domains including hotel booking, restaurant reservation, and taxi ordering. Each dialogue is annotated with the dialogue state at each turn. Keeping consistent with prior work, we use the 2.1 version of MultiWOZ (Eric et al., 2019). We also perform experiments on **SpokenWOZ** (Si et al., 2024), a spoken dialogue TOD dataset inspired by MultiWOZ. SpokenWOZ dialogues were collected from crowdworkers engaging in spoken conversations and includes text transcriptions from an automatic speech recognition system. We perform our experiments on the audio transcriptions.

For effective comparison with the prior work (D3ST and ESA), we choose `{taxi,train}` as our holdout domains to minimize slot overlap between the training and target domains. The slot overlap between domains is provided in Appendix A.4.

[1]Reimplementation from (Richardson et al., 2024).

Method	JGA$_{taxi}$	JGA$_{train}$	TGA
Prior work			
TransferQA	61.9	36.7	15.9
T5DST	64.6	35.4	15.7
Prompter	66.3	39.0	20.2
DualLoRA	67.2	*42.4*	24.2
D3ST	**78.4**	38.7	*25.2*
ESA	69.5	50.6	34.9
Our approach			
gFFS	*72.1*	*54.1*	*39.9*
pFFS	69.4	**56.4**	**40.7**

Table 2: Comparison of our method to prior work on MultiWOZ 2.1. Goal accuracies (%) showing JGA reported in original prior works with the corresponding estimated TGA.

4.5 Main Results

Our main results are shown in Table 2. Our best method, pFFS, improves TGA over the best performing baseline, ESA, by 5.8%. It also achieves the highest JGA on `train`. Our other feedback method gFFS achieves the second highest JGA on `taxi`.

4.6 SpokenWOZ Results

Method	JGA$_{taxi}$	JGA$_{train}$	TGA
Prior work			
D3ST [1]	67.1	41.1	21.9
ESA	66.7	50.5	30.3
Our approach			
gFFS	67.2	51.2	31.3
pFFS	**67.5**	**52.9**	**33.4**

Table 3: SpokenWOZ results. Comparison of our method to our re-implementation of D3ST, the highest performer on MultiWOZ. Note that JGA here is overall JGA on the full test set with all domains. TGA is the same metric as before, computed over our holdout domains {`taxi`,`train`}.

Table 3 shows the results for SpokenWOZ where we reimplement the best performing methods on MultiWOZ. The results demonstrate the advantages of few-shot feedback over the baselines, consistently outperforming prior work in terms of JGA and TGA by 2%.

5 Ablation Study

Because our few-shot feedback method builds on ESA, we wish to assess the impact of augmentation and feedback on the gains independently. To this end, we run three ablations by removing each component individually, and then removing both. To remove feedback, we run inference using the boostrapped model because the model was already trained to convergence on the original data. When removing ESA, we boostrap a model without the augmentation (standard fine-tuning), then collect feedback and refine the model with another training cycle, again without augmentation. To remove both we simply train untit convergence on the original data with no augmentation. We chose our best performing method, pFFS, for the ablation study.

Method	Sch. Aug.	Feedback	TGA	
			MWOZ	SWOZ
pFFS	✓	✓	40.7	33.4
-aug.	✗	✓	39.0	31.1
-feedback	✓	✗	34.9	30.3
-both	✗	✗	30.1	25.7

Table 4: Impact of Schema Augmentation and Feedback on our method, pFFS

Our ablation study results are shown in Table 4. While augmentation does bolster the effectiveness of feedback, it is less important than the feedback itself. TGA on MultiWOZ only degrades 40.7% $\rightarrow$ 39.0% when augmentation is removed, but degrades to 34.9% if augmentation is kept but feedback removed. We see a similar trend for SpokenWOZ as well. As expected, removing both degrades the results even more for both datasets. This ablation demonstrates the strength of FFS while highlighting the key role augmentation plays.

6 Conclusion

We developed a two-step approach for zero-shot domain adaptation in dialogue state tracking, consisting of 1) boostrapping a draft model using Schema Augmentation, then 2) Collecting few-shot language feedback from erroneous generations and training a final model using that feedback. We demonstrated superior domain adaptation performance of our method across two popular task-oriented dialogue datasets. In addition, we investigated the impact of data augmentation and feedback separately in an ablation study. Our results motivate further research into few-shot feedback and its potential for improving language modelling performance beyond existing training methods.

Limitations

Our work has a few limitations. The results would be strengthened by exploration of additional models. Additionally, while we tested on two task-oriented dialogue datasets (MultiWOZ and SpokenWOZ), both are based on similar domains, and further testing on more diverse datasets is needed. Additionally, we only used one set of holdout domains, whereas our experiments could be repeated using different sets of the available domains as holdouts. Due to compute constraints, we limited fine-tuning to models with fewer than 10 billion parameters, which may affect performance compared to larger models. Moreover, our experiments were confined to English-language datasets, leaving the effectiveness of our few-shot feedback method in multilingual or non-English contexts unexplored. Lastly, the scope of hyperparameter tuning was limited by available resources, and further exploration of fine-tuning configurations could yield even more insights.

Ethics Statement

This work aims to improve dialogue state tracking in task-oriented systems, with potential applications in real-world settings like customer service or healthcare. Ensuring the fairness and robustness of these models is crucial to avoid biased or harmful outcomes, especially for underrepresented groups. Additionally, while our method enhances model performance in unseen domains, careful consideration is required before deploying such models in sensitive areas where errors could have significant consequences. Finally, the environmental impact of training large models is an important factor, and more sustainable practices in AI research should be prioritized.

References

Taha Aksu, Min-Yen Kan, and Nancy F Chen. 2023. Prompter: Zero-shot adaptive prefixes for dialogue state tracking domain adaptation. *arXiv preprint arXiv:2306.04724*.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Inigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. Multiwoz–a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. *arXiv preprint arXiv:1810.00278*.

Mihail Eric, Rahul Goel, Shachi Paul, Adarsh Kumar, Abhishek Sethi, Peter Ku, Anuj Kumar Goyal, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: A consolidated multi-domain dialogue dataset with state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

Yujie Feng, Zexin Lu, Bo Liu, Liming Zhan, and Xiao-Ming Wu. 2023. Towards llm-driven dialogue state tracking. *arXiv preprint arXiv:2310.14970*.

Sanja Fidler et al. 2017. Teaching machines to describe images with natural language feedback. *Advances in Neural Information Processing Systems*, 30.

Gemma Team. 2024. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*.

Larry Heck, Simon Heck, and Anirudh S. Sundar. 2024. mForms : Multimodal form filling with question answering. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 11262–11271, Torino, Italia. ELRA and ICCL.

Michael Heck, Nurul Lubis, Benjamin Ruppik, Renato Vukovic, Shutong Feng, Christian Geishauser, Hsien-Chin Lin, Carel van Niekerk, and Milica Gašić. 2023. Chatgpt for zero-shot dialogue state tracking: A solution or an opportunity? *arXiv preprint arXiv:2306.01386*.

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.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Jiwei Li, Alexander H Miller, Sumit Chopra, Marc'Aurelio Ranzato, and Jason Weston. 2022. Dialogue learning with human-in-the-loop. In *International Conference on Learning Representations*.

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. *arXiv preprint arXiv:2101.08333*.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, et al. 2021a. Zero-shot dialogue state tracking via cross-task transfer. *arXiv preprint arXiv:2109.04655*.

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 state tracking. *arXiv preprint arXiv:2105.04222*.

Xiang Luo, Zhiwen Tang, Jin Wang, and Xuejie Zhang. 2024. Zero-shot cross-domain dialogue state tracking via dual low-rank adaptation. *arXiv preprint arXiv:2407.21633*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. 2024. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36.

Sabrina McCallum, Max Taylor-Davies, Stefano V Albrecht, and Alessandro Suglia. 2023. Is feedback all you need? leveraging natural language feedback in goal-conditioned reinforcement learning. *arXiv preprint arXiv:2312.04736*.

Baolin Peng, Michel Galley, Pengcheng He, Hao Cheng, Yujia Xie, Yu Hu, Qiuyuan Huang, Lars Liden, Zhou Yu, Weizhu Chen, et al. 2023. Check your facts and try again: Improving large language models with external knowledge and automated feedback. *arXiv preprint arXiv:2302.12813*.

Christopher Richardson, Roshan Sharma, Neeraj Gaur, Parisa Haghani, Anirudh Sundar, and Bhuvana Ramabhadran. 2024. Schema augmentation for zero-shot domain adaptation in dialogue state tracking. *arXiv preprint arXiv:2411.00150*.

Christopher Richardson, Anirudh Sundar, and Larry Heck. 2023. Syndicom: Improving conversational commonsense with error-injection and natural language feedback. In *Proceedings of the 24th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 297–308.

Jérémy Scheurer, Jon Ander Campos, Jun Shern Chan, Angelica Chen, Kyunghyun Cho, and Ethan Perez. 2022. Training language models with language feedback. *arXiv preprint arXiv:2204.14146*.

Shuzheng Si, Wentao Ma, Haoyu Gao, Yuchuan Wu, Ting-En Lin, Yinpei Dai, Hangyu Li, Rui Yan, Fei Huang, and Yongbin Li. 2024. Spokenwoz: A large-scale speech-text benchmark for spoken task-oriented dialogue agents. *Advances in Neural Information Processing Systems*, 36.

Manya Wadhwa, Xinyu Zhao, Junyi Jessy Li, and Greg Durrett. 2024. Learning to refine with fine-grained natural language feedback. *arXiv preprint arXiv:2407.02397*.

Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O'Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu, Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Shepherd: A critic for language model generation. *arXiv preprint arXiv:2308.04592*.

Jason E Weston. 2016. Dialog-based language learning. *Advances in Neural Information Processing Systems*, 29.

Zihao Yi, Jiarui Ouyang, Yuwen Liu, Tianhao Liao, Zhe Xu, and Ying Shen. 2024. A survey on recent advances in llm-based multi-turn dialogue systems. *arXiv preprint arXiv:2402.18013*.

Jeffrey Zhao, Raghav Gupta, Yuan Cao, Dian Yu, Mingqiu Wang, Harrison Lee, Abhinav Rastogi, Izhak Shafran, and Yonghui Wu. 2022. Description-driven task-oriented dialog modeling. *arXiv preprint arXiv:2201.08904*.

A Appendix

A.1 Example

An example prompt from the dataset is provided in Figure 2.

A.2 Hyperparameters

For our experiments, we fine-tune both our models using the AdamW optimizer with a learning rate of 2e-4 and warmup ratio of 0.03. Due to compute constraints, we use LoRA (Hu et al., 2021) to train adapters while keeping base weights frozen. We use a LoRA $r = 2$ and $\alpha = 2$ with a dropout of 0, and adapter weights added to all linear layers. In all experiments, the modes is fine-tuned to convergence in each experiment. We achieve this by evaluating on the validation split each epoch and choosing an early stopping patience of 1. This ensures that each experiment yields the best model and comparisons between methods are fair.

All experiments use a random seed of 42 and deterministic algorithms were used everywhere possible to ensure minimal variation between runs. All accuracy metrics reported had less than 1% variance across all runs.

A.3 Compute

All fine-tuning and inference was run on Nvidia A40 GPUs with 48GB GDDR6 memory. Fine-tuning took 1-2 hours on 8 GPUs in parallel with pytorch distributed data parallel (DDP).

A.4 Slots and Domains

Tables 5 and 6 show the domain and slot combinations for the two datasets. Taxi, train, and bus were chosen as holdout domains due to many slots in common with each other and few slots in common with other domains.

```
    {
    "dialogue_id": "SNG0548.json",
    "prompt": "### Instructions: Give the dialogue state at the end of the given
    ↪  dialogue, formatted in JSON. Follow the schema and only use the given
    ↪  pre-defined slots and their possible values. `Possible values: []` means
    ↪  open-ended (the slot can take on any value). Omit any slots with empty
    ↪  values from your answer. If no slots can be filled from the dialogue,
    ↪  respond with an empty json object.\n\n### Schema: \n- Slot:
    ↪  restaurant-pricerange; Description: price budget for the restaurant;
    ↪  Possible values: ['cheap', 'expensive', 'moderate']\n- Slot:
    ↪  restaurant-area; Description: area or place of the restaurant; Possible
    ↪  values: ['centre', 'east', 'north', 'south', 'west']\n- Slot:
    ↪  restaurant-food; Description: the cuisine of the restaurant you are looking
    ↪  for; Possible values: []\n- Slot: restaurant-name; Description: name of the
    ↪  restaurant; Possible values: []\n- Slot: restaurant-bookday; Description:
    ↪  day of the restaurant booking; Possible values: ['monday', 'tuesday',
    ↪  'wednesday', 'thursday', 'friday', 'saturday', 'sunday']\n- Slot:
    ↪  restaurant-bookpeople; Description: how many people for the restaurant
    ↪  reservation; Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']\n-
    ↪  Slot: restaurant-booktime; Description: time of the restaurant booking;
    ↪  Possible values: []\n- Slot: restaurant-address; Description: address of
    ↪  the restaurant; Possible values: []\n- Slot: restaurant-phone; Description:
    ↪  phone number of the restaurant; Possible values: []\n- Slot:
    ↪  restaurant-postcode; Description: postal code of the restaurant; Possible
    ↪  values: []\n- Slot: restaurant-ref; Description: reference number of the
    ↪  restaurant booking; Possible values: []\n\n\n### Dialogue: \nUSER: i am
    ↪  looking for a particular restaurant . it is called pizza hut city centre
    ↪  .\n\n",
    "answer": {
    "restaurant-name": "pizza hut city centre"
  },
}
```

Figure 2: Example Prompt in the dataset

	attraction	bus	hospital	hotel	police	restaurant	taxi	train
area	✓			✓		✓		
arriveby							✓	✓
bookday				✓		✓		
bookpeople				✓		✓		✓
bookstay				✓				
booktime						✓		
day		✓						✓
department			✓					
departure		✓					✓	✓
destination		✓					✓	✓
food						✓		
internet				✓				
leaveat		✓					✓	✓
name	✓			✓	✓	✓		
parking				✓				
pricerange				✓		✓		
stars				✓				
type	✓			✓				

Table 5: Domain-Slot Combinations for MultiWOZ

	attraction	**hospital**	**hotel**	**profile**	**restaurant**	**taxi**	**train**
area	✓		✓		✓		
arriveby						✓	✓
day			✓		✓		✓
department		✓					
departure						✓	✓
destination						✓	✓
email				✓			
food					✓		
idnumber				✓			
internet			✓				
leaveat						✓	✓
name	✓		✓	✓	✓		
parking			✓				
people			✓		✓		✓
phonenumber				✓			
platenumber				✓			
pricerange			✓				
stars			✓				
stay			✓				
time					✓		
type	✓		✓				

Table 6: Domain-Slot Combinations for SpokenWOZ.

A.5 Replacements

The full list of synonym and encoding replacements for slots and values is shown in Figure 3.

A.6 Licenses

All datasets and models used are free and open-source. MultwiWOZ and SpokenWOZ are available under MIT license. Gemma-2 is available under the Apache 2.0 license. All models and datasets are intended for reseach purposes, which is consistent with this work. Datasets are widely used and reported not to contain personal information or offensive content.

```
{
    "slots": {
        "pricerange": ["slot010", "slot011", "slot012", "slot013", "slot014"]
        "type": ["slot020", "slot021", "slot022", "slot023", "slot024"]
        "parking": ["slot030", "slot031", "slot032", "slot033", "slot034"]
        "day": ["slot040", "slot041", "slot042", "slot043", "slot044"]
        "bookday": ["slot050", "slot051", "slot052", "slot053", "slot054"]
        "people": ["slot060", "slot061", "slot062", "slot063", "slot064"],
        "bookpeople": ["slot070", "slot071", "slot072", "slot073", "slot074"],
        "stay": ["slot080", "slot081", "slot082", "slot083", "slot084"],
        "bookstay": ["slot090", "slot091", "slot092", "slot093", "slot094"],
        "internet": ["slot100", "slot101", "slot102", "slot103", "slot104"],
        "name": ["slot110", "slot111", "slot112", "slot113", "slot114"],
        "area": ["slot120", "slot121", "slot122", "slot123", "slot124"],
        "stars": ["slot130", "slot131", "slot132", "slot133", "slot134"],
        "arriveby": ["slot140", "slot141", "slot142", "slot143", "slot144"],
        "leaveat": ["slot150", "slot151", "slot152", "slot153", "slot154"],
        "destination": ["slot160", "slot161", "slot162", "slot163", "slot164"],
        "departure": ["slot170", "slot171", "slot172", "slot173", "slot174"],
        "food": ["slot180", "slot181", "slot182", "slot183", "slot184"],
        "time": ["slot190", "slot191", "slot192", "slot193", "slot194"],
        "booktime": ["slot200", "slot201", "slot202", "slot203", "slot204"],
        "department": ["slot210", "slot211", "slot212", "slot213", "slot214"],
        "email": ["slot220", "slot221", "slot222", "slot223", "slot224"],
        "idnumber": ["slot230", "slot231", "slot232", "slot233", "slot234"],
        "phone": ["slot240", "slot241", "slot242", "slot243", "slot244"],
        "phonenumber": ["slot250", "slot251", "slot252", "slot253", "slot254"],
        "platenumber": ["slot260", "slot261", "slot262", "slot263", "slot264"],
        "address": ["slot270", "slot271", "slot272", "slot273", "slot274"],
        "postcode": ["slot280", "slot281", "slot282", "slot283", "slot284"],
        "ref": ["slot290", "slot291", "slot292", "slot293", "slot294"],
        "entrancefee": ["slot300", "slot301", "slot302", "slot303", "slot304"],
        "openhours": ["slot310", "slot311", "slot312", "slot313", "slot314"]
    },
    "domains": {
        "attraction": ["domain10", "domain11", "domain12", "domain13", "domain14"],
        "hotel": ["domain20", "domain21", "domain22", "domain23", "domain24"],
        "hospital": ["domain30", "domain31", "domain32", "domain33", "domain34"],
        "police": ["domain40", "domain41", "domain42", "domain43", "domain44"],
        "profile": ["domain50", "domain51", "domain52", "domain53", "domain54"],
        "restaurant": ["domain60", "domain61", "domain62", "domain63", "domain64"],
        "taxi": ["domain70", "domain71", "domain72", "domain73", "domain74"],
        "train": ["domain80", "domain81", "domain82", "domain83", "domain84"]
    }
}
```

Figure 3: Domain and slot synonym replacements for data augmentation

```
### Schema:
- Slot: hotel-pricerange; Description: price budget of the hotel; Possible values:
↪  ['expensive', 'cheap', 'moderate']
- Slot: hotel-type; Description: what is the type of the hotel; Possible values:
↪  ['guesthouse', 'hotel']
- Slot: hotel-parking; Description: whether the hotel has parking; Possible values:
↪  ['free', 'no', 'yes']
- Slot: hotel-bookday; Description: day of the hotel booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: hotel-bookpeople; Description: number of people for the hotel booking;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: hotel-bookstay; Description: length of stay at the hotel; Possible values:
↪  ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: hotel-stars; Description: star rating of the hotel; Possible values: ['0',
↪  '1', '2', '3', '4', '5']
- Slot: hotel-internet; Description: whether the hotel has internet; Possible
↪  values: ['free', 'no', 'yes']
- Slot: hotel-name; Description: name of the hotel; Possible values: []
- Slot: hotel-area; Description: area or place of the hotel; Possible values:
↪  ['centre', 'east', 'north', 'south', 'west']
- Slot: hotel-address; Description: address of the hotel; Possible values: []
- Slot: hotel-phone; Description: phone number of the hotel; Possible values: []
- Slot: hotel-postcode; Description: postal code of the hotel; Possible values: []
- Slot: hotel-ref; Description: reference number of the hotel booking; Possible
↪  values: []
- Slot: restaurant-pricerange; Description: price budget for the restaurant;
↪  Possible values: ['cheap', 'expensive', 'moderate']
- Slot: restaurant-area; Description: area or place of the restaurant; Possible
↪  values: ['centre', 'east', 'north', 'south', 'west']
- Slot: restaurant-food; Description: the cuisine of the restaurant you are looking
↪  for; Possible values: []
- Slot: restaurant-name; Description: name of the restaurant; Possible values: []
- Slot: restaurant-bookday; Description: day of the restaurant booking; Possible
↪  values: ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday',
↪  'sunday']
- Slot: restaurant-bookpeople; Description: how many people for the restaurant
↪  reservation; Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: restaurant-booktime; Description: time of the restaurant booking; Possible
↪  values: []
- Slot: restaurant-address; Description: address of the restaurant; Possible
↪  values: []
- Slot: restaurant-phone; Description: phone number of the restaurant; Possible
↪  values: []
- Slot: restaurant-postcode; Description: postal code of the restaurant; Possible
↪  values: []
- Slot: restaurant-ref; Description: reference number of the restaurant booking;
↪  Possible values: []
```

Figure 4: Original data sample schema

```
### Schema:
- Slot: domain2-slot1; Description: price budget of the hotel; Possible values:
↪  ['expensive', 'cheap', 'moderate']
- Slot: domain2-slot2; Description: what is the type of the hotel; Possible values:
↪  ['guesthouse', 'hotel']
- Slot: domain2-slot3; Description: whether the hotel has parking; Possible values:
↪  ['free', 'no', 'yes']
- Slot: domain2-slot5; Description: day of the hotel booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: domain2-slot7; Description: number of people for the hotel booking;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: domain2-slot9; Description: length of stay at the hotel; Possible values:
↪  ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: domain2-slot13; Description: star rating of the hotel; Possible values:
↪  ['0', '1', '2', '3', '4', '5']
- Slot: domain2-slot10; Description: whether the hotel has internet; Possible
↪  values: ['free', 'no', 'yes']
- Slot: domain2-slot11; Description: name of the hotel; Possible values: []
- Slot: domain2-slot12; Description: area or place of the hotel; Possible values:
↪  ['centre', 'east', 'north', 'south', 'west']
- Slot: domain2-slot27; Description: address of the hotel; Possible values: []
- Slot: domain2-slot24; Description: phone number of the hotel; Possible values: []
- Slot: domain2-slot28; Description: postal code of the hotel; Possible values: []
- Slot: domain2-slot29; Description: reference number of the hotel booking;
↪  Possible values: []
- Slot: domain6-slot1; Description: price budget for the restaurant; Possible
↪  values: ['cheap', 'expensive', 'moderate']
- Slot: domain6-slot12; Description: area or place of the restaurant; Possible
↪  values: ['centre', 'east', 'north', 'south', 'west']
- Slot: domain6-slot18; Description: the cuisine of the restaurant you are looking
↪  for; Possible values: []
- Slot: domain6-slot11; Description: name of the restaurant; Possible values: []
- Slot: domain6-slot5; Description: day of the restaurant booking; Possible values:
↪  ['monday', 'tuesday', 'wednesday', 'thursday', 'friday', 'saturday', 'sunday']
- Slot: domain6-slot7; Description: how many people for the restaurant reservation;
↪  Possible values: ['1', '2', '3', '4', '5', '6', '7', '8']
- Slot: domain6-slot20; Description: time of the restaurant booking; Possible
↪  values: []
- Slot: domain6-slot27; Description: address of the restaurant; Possible values: []
- Slot: domain6-slot24; Description: phone number of the restaurant; Possible
↪  values: []
- Slot: domain6-slot28; Description: postal code of the restaurant; Possible values:
↪  []
- Slot: domain6-slot29; Description: reference number of the restaurant booking;
↪  Possible values: []
```

Figure 5: Example schema for data augmentation