Parameter-Efficient Low-Resource Dialogue State Tracking by Prompt Tuning

Anonymous Author(s)

Abstract

Dialogue state tracking (DST) is an important step in dialogue management to keep track of users' beliefs. Existing works fine-tune all language model (LM) parameters to tackle the DST task, which requires significant data and computing resources for training and hosting. The cost grows exponentially in the real-world deployment where dozens of fine-tuned LM are used for different domains and tasks. To reduce parameter size and better utilize cross-task shared information, we propose to use soft prompt token embeddings to learn task properties. Without tuning LM parameters, our method drastically reduces the number of parameters needed to less than 0.5% of prior works while achieves better low-resource DST performance.

1 Introduction

Dialogue state tracking (DST) that extracts structured conversation progress in a list of slot-value pairs from unstructured dialogue utterances is an essential component of a dialogue system (Wang and Lemon, 2013). Unlike classification-based models that pick the slot value from given candidate (Ye et al., 2021; Chen et al., 2020), recent works formulate DST as a conditional generation task (Gao et al., 2019; Lin et al., 2020), where the concatenation of dialogue history and a slot-specific prompt are fed to generative models and the text generation output are decoded to predicted slot values (Ham et al., 2020; Hosseini-Asl et al., 2020). This formulation enjoys the benefit of generalizability to unseen domains and slot types beyond a defined dialogue ontology (Li et al., 2021; Peng et al., 2021).

General prompting methods use a textual prompt to provide task information to the LM (Liu et al., 2021; Gao et al., 2021). Prior works have variations that update different parameter combinations such as both LM and prompt token embeddings (Gao et al., 2021; Li and Liang, 2021), only the token embeddings of the LM (Zhu et al., 2021), or only the prompt token embeddings (Lester et al., 2021; Gu et al., 2022; Vu et al., 2022).

While there are some existing prompt-based approaches for DST with different designs of prompts such as using slot name (Lee and Jha, 2019; Zhao et al., 2021; Lee et al., 2021; Su et al., 2022), slot description (Rastogi et al., 2020), slot type (Lin et al., 2021b), possible values (Lin et al., 2021b), priming examples (Gupta et al., 2022) and/or slot-specific question (Gao et al., 2019; Zhou and Small, 2019; Gao et al., 2020; Lin et al., 2021a; Li et al., 2021) in prompt sentences, they all fine-tune the entire LM along with the prompt tokens for a new domain, which requires a significant amount of training time, system resources, and annotated data (Clarke et al., 2022; Sauer et al., 2022). The computing and data resource-hungry issues are more severe in the real-world deployment where LMs tuned for different domains and tasks need to be trained and hosted, and a typical dialogue system has to serve dozens of such LMs (Maronikolakis and Schütze, 2021; Strubell et al., 2019; Lacoste et al., 2019). This leads to a high cost of the development and service of dialogue systems and constrains offline deployment. In addition, limited data is available for a new domain or task.

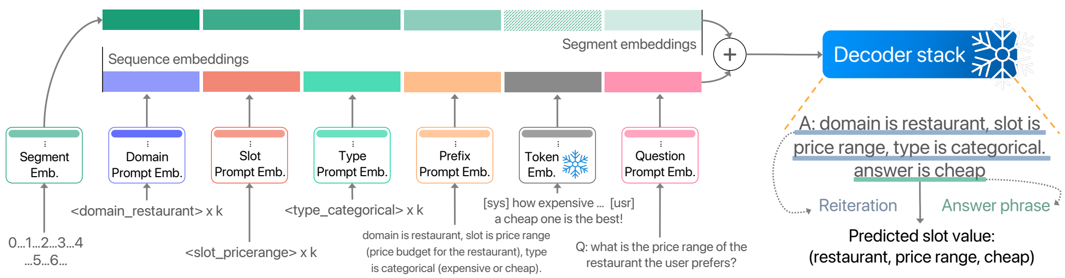

Figure 1: Model design. The snow icon indicates non-trainable parameters. Absolute positional embeddings are added together with segment embeddings and sequence embeddings, we omit it for simplicity in the illustration.

We propose a **parameter-efficient** and **data-efficient** DST model for **low-resource** settings, which only needs to update 0.08% of parameters compared with the previous best model, by keeping LM parameters frozen and introducing soft prompt tokens to represent task properties of different slots. Fig. 1 gives an overview of our model. The only prior work we are aware of that only updates prompt token embeddings and thus parameter-efficient is Zhu et al. (2022), but it focuses on continual domain adaptation and with a significant amount of training data.

Our design introduces three techniques that are generalizable to other generative-based information extraction models. 1) **Task-specific parameters**: *task prompt tokens* are introduced to specifically learn domain, slot and slot type information so that the model behaves according to the task; *word-mapping prompt tokens* enable us to obtain task knowledge contained in natural language instruction and optimize human-created prompts with continuous embedding space. 2) **Task metadata in objective**: we introduce the reiteration technique in the target sequence in order to include explicit task signals in the text generation objective. 3) **Distinguishing segments**: segment embeddings help the model identify the prompt segment, dialogue speakers, and question partition. Our proposed method enables much more efficient dialogue system deployment as only one LM needs to be hosted and inference for different domains could be realized by feeding domain-specific prompt token embeddings into the transformer stack.

Experiments on MultiWOZ 2.0 show that our method achieves better performance on low-resource DST with orders of magnitude fewer parameters. We further conduct ablation studies, error analysis, and examine the semantic information shown in the prompt tokens. We observe that our model is more specialized in predicting categorical slot values, is more conservative for slots with free output space and introduces more hallucination errors for categorical slots.

2 Method

2.1 Task Definition

The goal is to construct a belief state with $|S|$ pairs of slot and value at a certain turn in a multi-turn conversation. All the turns up to the query turn are dialogue history, and slot-specific information (*i.e.* name, description, value candidates, question and type of the slot) is provided.[1]

2.2 Generative Seq2seq Framework

We use a decoder-only pre-trained language model (PLM) GPT-2 (Radford et al., 2019) as the backbone to provide language and commonsense knowledge, rather than an encoder-decoder model because of its superior performance (Li et al., 2021). To get a belief state at a certain turn, we create $|S|$ data instances to predict the slot value for each slot. Fig. 1 demonstrates the design and a sample query.

Input sequence. We construct the input sequence by concatenating the following segments: 1) *Task prompt tokens for domain, slot and type*, each has k prompt tokens and they are shared among instances with the same domain, slot or type; 2) *Prefix*, a short sentence containing slot description,

[1]We show slot-specific info in App. A.1 and App. A.2.

names of domain, slot, and type, and all possible candidates if the query slot is categorical; 3) *Dialogue history*, in which [sys] and [usr] tokens are used to indicate the speaker; and 4) *Question*, human-written question about the slot.

Target sequence and reiteration. We introduce the reiteration technique in the target sequence as shown in Fig. 1 and generate task information before the answer phrase. This technique allows the model to optimize upon both the answer and the sentence containing slot metadata, and explicitly learn the task information.

Segment embeddings. The input sequence contains segments with diverse formats and they are quite different from the format used in the pre-training phase of the LM. We divide the input sequence into segments, including five prompt segments, the system turns, the user turns and the answer segment. Tokens within a segment are assigned the same segment ID. Segment embeddings, which have the same length as the input sequence, are added with sequence embeddings and positional embeddings. We randomly initialize the embeddings of segment IDs and update them during training.

Training and inference. We pass the combined embeddings to the decoder stack to calculate the likelihood over the vocabulary. We use the cross-entropy loss with a regularization term to constrain the scale of prompt token embeddings following $L = CE + \|PE' - PE\|_2^2$ where PE' and PE are updated and initialized prompt token embeddings (Müller et al., 2022). Parameters of the PLM are frozen, and only prompt and segment embeddings are updated with Adam optimizer. During inference, we generate the output autoregressively with greedy decoding, and extract the answer with a rule-based function.

2.3 Soft Prompt Tokens

Prompt segments. We use two kinds of prompt tokens. *Task prompt tokens* are chosen according to the task's metadata, and used in the domain, slot and type prompt segments. *Word-mapping prompt tokens* are mapped from existing tokens in the prefix and question parts and used to replace normal tokens. In other words, task and word-mapping prompt tokens are shared across instances with the same task and instances using the same words respectively. We concatenate embeddings of each prompt segment (obtained by separate embedding matrices) with dialogue history embeddings (obtained by the frozen token embedding matrix) to form sequence embeddings.

Prompt initialization. To boost the performance in the low-resource setting, we use the pre-trained token embeddings to initialize the soft prompt token embeddings. The token embeddings from PLM are used to represent word semantics for language understanding, while the soft prompt tokens are used to represent task information initialized by task-related semantic meanings. We initialize a task prompt token by embedding of a randomly chosen token from its domain, slot or slot type name. Word-mapping prompt tokens are initialized with the embedding of the mapped word.

3 Experimental Setup

Dataset. We experiment on dialogues of five domains (*i.e*. attraction, hotel, restaurant, train, taxi) in MultiWOZ 2.0 (Budzianowski et al., 2018).

Settings. We evaluate using the low-resource few-shot DST task. We take 5, 10, 20, 1%, 5% and 10% of training conversations to train, and evaluate on the full test set of each target domain.[2]

Evaluation metrics. Joint Goal Accuracy (JGA) represents the proportion of *turns* with *all* slots predicted correctly, and Slot Accuracy (SA) reflects the proportion of correct *slots*. If a slot is empty at a certain turn (for example, no related information is mentioned), the model needs to predict "none".

Baseline models. We compare with the following works. 1) TRADE (Wu et al., 2019): GRU-based model with copy mechanism; 2) DSTQA (Zhou and Small, 2019): QA-style model using ELMo representation; 3) T5DST (Lin et al., 2021b): T5-based generative model with slot type as prompt; 4) Lee et al. (2021): T5-based generative model with slot description and possible slot values as prompt; 5) Li et al. (2021): GPT-2 based QA-style generative model with manually created questions. The entire

[2]App. C.3 and C.4 show experimental setting details.

language model is updated for T5DST, Lee et al. and Li et al., and they represents the performance of prompt-based DST works. App. B.6 shows comparison with baselines' frozon LM variation.

4 Experimental Results

Table 1: Overall performance. Detailed parameter counts are in App. A.3, variances are in App. B.5.

Model	Params#	5	10	20	1%	5%	10%	5	10	20	1%	5%	10%	5	10	20	1%	5%	10%
		Attraction (3 slots, 1% = 27 conv.)						Hotel (10 slots, 1% = 33 conv.)						Restaurant (7 slots, 1% = 38 conv.)					
TRADE		—	—	—	—	52.19	58.46	—	—	—	—	31.93	41.29	—	—	—	—	47.31	53.65
DSTQA		—	—	—	—	51.58	61.77	—	—	—	—	33.08	**49.69**	—	—	—	—	35.33	54.27
T5DST	60M	4.77	21.93	30.57	40.68	52.12	60.13	8.19	13.46	17.94	18.63	38.76	46.13	13.80	19.51	22.79	29.47	**53.32**	58.44
Lee et al.	60M	6.33	19.12	34.53	37.56	54.34	58.75	9.31	15.76	22.07	24.41	**40.11**	42.98	15.87	19.66	22.15	30.96	48.94	**58.59**
Li et al.	335M	7.90	27.09	35.63	42.18	49.13	60.85	12.49	15.15	19.44	24.04	37.88	46.47	17.27	22.30	25.68	30.70	49.75	58.50
Ours	271K	**33.56**	**39.41**	**45.75**	**47.28**	**56.99**	**63.61**	**15.63**	**18.18**	**22.50**	**33.01**	38.24	45.60	**19.76**	**25.72**	**27.65**	**34.40**	50.81	55.79
		Taxi (4 slots, 1% = 15 conv.)						Train (6 slots, 1% = 29 conv.)						Average					
TRADE		—	—	—	—	59.03	60.51	—	—	—	—	48.82	59.65	—	—	—	—	47.86	54.71
DSTQA		—	—	—	—	58.25	59.35	—	—	—	—	50.36	61.28	—	—	—	—	45.72	57.27
T5DST	60M	48.22	53.74	58.27	58.19	59.23	69.03	12.31	21.93	36.45	43.93	69.27	69.48	17.46	26.11	33.20	38.18	54.54	60.64
Lee et al.	60M	45.32	49.93	58.58	58.52	60.77	**71.23**	13.57	25.02	38.52	50.26	69.32	69.72	18.08	25.90	35.17	40.34	54.70	60.25
Li et al.	335M	50.99	57.47	58.49	58.26	**61.68**	69.23	17.56	27.42	39.27	45.32	**71.69**	73.45	21.24	29.89	35.70	40.10	54.03	**61.70**
Ours	271K	**51.11**	**59.63**	**60.89**	**60.33**	61.63	63.00	**18.95**	**30.95**	**50.34**	**52.05**	69.51	**75.00**	**27.80**	**34.78**	**41.43**	**45.41**	**55.44**	60.60

Overall results. We show the overall few-shot experimental results in Table 1. Although our model uses only 0.08% and 0.45% of parameters compared with baselines, it still achieves higher JGA than all baseline models when using 1% or less training data across all domains. Especially we observe around 5, and 9 points JGA increases for the `attraction` and `hotel` domains compared with existing best models with 1% training data. In the `attraction` domain with 3 unique slots, our model trained using 5 dialogues performs on par with the previous best model using 20 dialogues. Our model shows its superiority especially when the amount of unique tasks is small. Using 5% and 10% data, our model performs comparably with existing best models with small gaps.

We demonstrate the performance of slots with different types in Fig. 2. We observe the worst performance in OPEN slots, which could be explained by the larger output candidate space.[3] Breaking down slot type to more fine-grained type lead to better result (considering DAY as a separate type rather than CATEGORICAL type, NUMBER and TIME as separate types rather than OPEN type). Compared with baselines, our model performs comparably on OPEN and TIME slots, but is more superior for CATEGORICAL, NUMBER and DAY slots.[4]

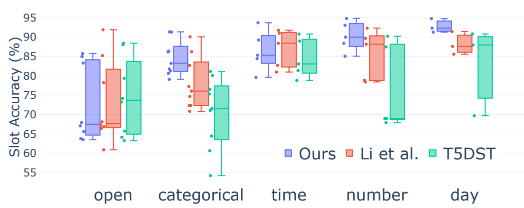

Figure 2: Slot accuracies across slot types using 1% training data, each dot represents a unique slot.

Ablation study. In Table 2, removing the slot segment (Line 2) leads to the largest performance drop among the three task prompt segments (L1-3), as slot is the most fine-grained task categorization. Prefix (L5) is more important than the question prompt (L4), which contains more metadata and parameters. The model without segment embedding (L6) has on average 7.8 points JGA drop, indicating the effectiveness of the segment embedding. We also observe an almost 2 points JGA drop (and an even larger drop with fewer training dialogues shown in

Table 2: Ablation study using 1% training data (JGA).

#	Model	Attr.	Hotel	Rest.	Taxi	Train	Avg
1	w/o domain	44.22	28.16	29.78	60.27	50.01	42.49
2	w/o slot	46.64	26.55	24.35	51.11	45.11	38.75
3	w/o type	45.30	25.26	33.65	59.89	51.91	43.20
4	w/o question	45.08	32.26	33.30	59.63	51.60	44.37
5	w/o prefix	42.98	28.78	31.54	57.72	47.00	41.60
6	w/o segment emb.	34.35	23.18	27.33	59.69	43.30	37.57
7	w/o reiteration	45.08	27.57	33.48	59.89	51.08	43.42
8	Full model	47.28	33.01	34.40	60.33	52.05	45.41

[3] A SA vs ontology size analysis is in App. B.2.

[4] SA for each slot and comparisons are in App. B.3.

App. B.1) without reiteration (L7), which shows the helpfulness of including explicit task information in the learning objective.

Error and qualitative analysis. We categorize error cases as: 1) hallucination: predicting value for an empty slot; 2) omission: predicting "none" for a non-empty slot; 3) wrong value: predicting wrong real value for a non-empty slot (Gao et al., 2020). Fig. 3 shows the error distribution in terms of the proportion of each error category. The general OPEN slots (including TIME and NUMBER) have relatively more omission errors, while the general CATEGORICAL slots have relatively more hallucination errors. Our model is more conservative for OPEN slots compared with Li et al..[5]

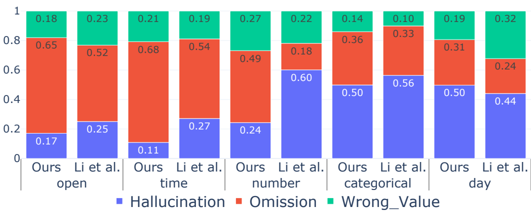

Figure 3: Error distribution across slot types

We then investigate semantic information contained in the learned prompt tokens by selecting the most changed prompt tokens and producing the closest tokens with the smallest cosine similarity between the learned prompt token embedding and frozen token embeddings of the PLM. We show the result for the `attraction` domain in Table 3, and for all domains in App. B.4. The closest tokens are mostly variations or semantically similar tokens of the expected meanings of prompt tokens.

Table 3: Closest tokens for the most changed prompt tokens in five prompt segments for the attraction domain.

Prompt token	Closest tokens
<domain_attraction_4>	raction; ractions; racted
<slot_name_2>	name; Name; names
<type_open_3>	open; Open; opened
special	special; Special; statistical
Q	answer; Answer; answered

5 Conclusion and Future Work

We propose a parameter-efficient DST model using prompt tuning, and it represents tasks with soft prompt tokens with segment awareness and reiteration. Our model achieves state-of-the-art low-resource DST performance with less than 0.5% parameters compared with fine-tuning LM. We plan to further investigate prompt aggregation.

References

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.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7521–7528.

Christopher Clarke, Joseph Peper, Karthik Krishnamurthy, Walter Talamonti, Kevin Leach, Walter Lasecki, Yiping Kang, Lingjia Tang, and Jason Mars. 2022. One agent to rule them all: Towards multi-agent conversational AI. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3258–3267, Dublin, Ireland. 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, Online. Association for Computational Linguistics.

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*

[5] Our model produces *relatively* larger proportion of omission error than Li et al., but it generate a reasonable amount of not-none values for non-empty slots as explained in App. B.7.

SIGdial Meeting on Discourse and Dialogue, pages 264–273, Stockholm, Sweden. Association for Computational Linguistics.

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.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. 2022. PPT: Pre-trained prompt tuning for few-shot learning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8410–8423, Dublin, Ireland. 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.

Donghoon Ham, Jeong-Gwan Lee, Youngsoo Jang, and Kee-Eung Kim. 2020. End-to-end neural pipeline for goal-oriented dialogue systems using GPT-2. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 583–592, Online. 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. *Advances in Neural Information Processing Systems*, 33:20179–20191.

Alexandre Lacoste, Alexandra Luccioni, Victor Schmidt, and Thomas Dandres. 2019. Quantifying the carbon emissions of machine learning. *arXiv preprint arXiv:1910.09700*.

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.

Sungjin Lee and Rahul Jha. 2019. Zero-shot adaptive transfer for conversational language understanding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6642–6649.

Brian Lester, Rami Al-Rfou, and Noah Constant. 2021. The power of scale for parameter-efficient prompt tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

Xiang Lisa Li and Percy Liang. 2021. Prefix-tuning: Optimizing continuous prompts for generation. 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 4582–4597, 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.

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, Online. Association for Computational Linguistics.

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*.

Antonis Maronikolakis and Hinrich Schütze. 2021. Multidomain pretrained language models for green NLP. In *Proceedings of the Second Workshop on Domain Adaptation for NLP*, pages 1–8, Kyiv, Ukraine. Association for Computational Linguistics.

Thomas Müller, Guillermo Pérez-Torró, and Marc Franco-Salvador. 2022. Few-shot learning with Siamese networks and label tuning. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8532–8545, Dublin, Ireland. Association for Computational Linguistics.

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.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

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.

Anna Sauer, Shima Asaadi, and Fabian Küch. 2022. Knowledge distillation meets few-shot learning: An approach for few-shot intent classification within and across domains. In *Proceedings of the 4th Workshop on NLP for Conversational AI*, pages 108–119, Dublin, Ireland. Association for Computational Linguistics.

Emma Strubell, Ananya Ganesh, and Andrew McCallum. 2019. Energy and policy considerations for deep learning in NLP. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3645–3650, Florence, Italy. 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.

Tu Vu, Brian Lester, Noah Constant, Rami Al-Rfou', and Daniel Cer. 2022. SPoT: Better frozen model adaptation through soft prompt transfer. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5039–5059, Dublin, Ireland. Association for Computational Linguistics.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *Proceedings of the SIGDIAL 2013 Conference*, pages 423–432, Metz, France. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, 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.

Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot self-attentive dialogue state tracking. In *Proceedings of the Web Conference 2021*, pages 1598–1608.

Yang You, Jing Li, Sashank Reddi, Jonathan Hseu, Sanjiv Kumar, Srinadh Bhojanapalli, Xiaodan Song, James Demmel, Kurt Keutzer, and Cho-Jui Hsieh. 2020. Large batch optimization for deep learning: Training bert in 76 minutes. 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, Online. Association for Computational Linguistics.

Jeffrey Zhao, Mahdis Mahdieh, Ye Zhang, Yuan Cao, and Yonghui Wu. 2021. Effective sequence-to-sequence dialogue state tracking. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7486–7493, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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

Qi Zhu, Bing Li, Fei Mi, Xiaoyan Zhu, and Minlie Huang. 2022. Continual prompt tuning for dialog state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1124–1137, Dublin, Ireland. Association for Computational Linguistics.

Yaoming Zhu, Jiangtao Feng, Chengqi Zhao, Mingxuan Wang, and Lei Li. 2021. Counter-interference adapter for multilingual machine translation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 2812–2823, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Checklist

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes]
   (c) Did you discuss any potential negative societal impacts of your work? [N/A]
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [Yes]
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]

(c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [Yes]

(d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes]

4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes]

(b) Did you mention the license of the assets? [N/A]

(c) Did you include any new assets either in the supplemental material or as a URL? [N/A]

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A]

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A]

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]

A Design Details

A.1 Slot Type Definitions

The slot types are defined according to output space and the number of possible answers. The slot types are defined in Table 4.

Table 4: Slot type definitions

Slot Types	Slots
Categorical	attraction-area, hotel-area, hotel-internet, hotel-parking, hotel-price range, hotel-type, restaurant-area, restaurant-price range, train-day
Day	hotel-book day, restaurant-book day
Number	hotel-book people, hotel-book stay, hotel-stars, restaurant-book people, train-book people
Open	attraction-open, attraction-type, hotel-name, restaurant-food, taxi-departure, taxi-destination, train-departure, train-destination
Time	restaurant-book time, taxi-arrive by, taxi-leave at, train-leave at

A.2 Question Prompt and Description

We show questions (used as question prompt) and description (as part of prefix prompt) for each slot in Table 12. Slot descriptions are from MultiWOZ 2.2 dataset (Zang et al., 2020).

A.3 Detailed Parameter Count

The average parameter count across all domains is 271K. We show detailed parameter count for each domain in Table 5. The parameters needed for each domain vary because the question and prefix prompt can map to a different set of prompt tokens for each domain. The parameters needed for each domain are calculated by adding prompt token embedding size ($prompt\ token\ count \times 1024$) with segment embedding size (8×1024 given 8 segments).

Table 5: Number of parameters needed for each domain. We list number of prompt tokens needed for each prompt segments, all prompt tokens needed and the ultimate parameter count.

	Attr.	Hotel	Rest.	Taxi	Train
Domain	5	20	20	10	10
Slot	15	200	140	40	60
Type	10	80	100	20	40
Question	20	46	36	19	27
Prefix	60	117	84	29	76
All	110	463	380	118	213
Params #	120832	482304	397312	129024	226304

B Additional Experimental Results

B.1 Additional Ablation Study for Reiteration

We show an additional ablation study to investigate the effect of introducing the reiteration technique in Table 6. We observe that the reiteration technique can lead to a significant increase in performance, especially with fewer amount of training dialogues. When there are limited training data, reiteration can help the model learn task boundaries among each slot faster and better.

Table 6: Ablation study for the reiteration technique.

Few-shot	Model	Attr.	Hotel	Rest.	Taxi	Train	Avg
5	w/o reit.	22.16	12.09	16.67	47.68	4.97	20.71
	w/ reit.	33.56	15.63	19.76	51.11	18.95	27.80
10	w/o reit.	23.08	12.39	13.75	56.39	9.26	22.97
	w/ reit.	39.41	18.18	24.72	59.63	30.95	34.58
1%	w/o reit.	45.08	27.57	33.48	59.89	51.08	43.42
	w/ reit.	47.28	33.01	34.40	60.33	52.05	45.41

B.2 Performance vs Ontology Size

We investigate the relationship between performance and the number of unique candidate answers (ontology size) using 1% target domain training data and Fig. 4 demonstrates the result with trendlines created by expanding average algorithm for each model. We also show the performance of two generative baseline models for comparison. We observe that the performance of all three models drops when the ontology size grows. For most ontology size, our model outperforms Li et al. and T5DST (Lin et al., 2021b).

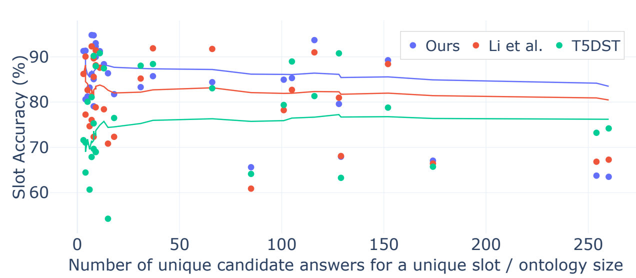

Figure 4: Performance for slots with different ontology sizes

B.3 Performance by Slots

To better understand the pros and cons of the prompt tuning method compared with fine-tuning LM, we show the slot accuracy difference for all unique slots training with 1% target domain data in Fig. 5. Our model outperforms Li et al. (2021) the most in CATEGORICAL-type "area" slots, and NUMBER-type slots "book people", all with at least 10% higher accuracy. Our model falls behind in 9 out of 30 slots, especially for the "restaurant: book time" slots.

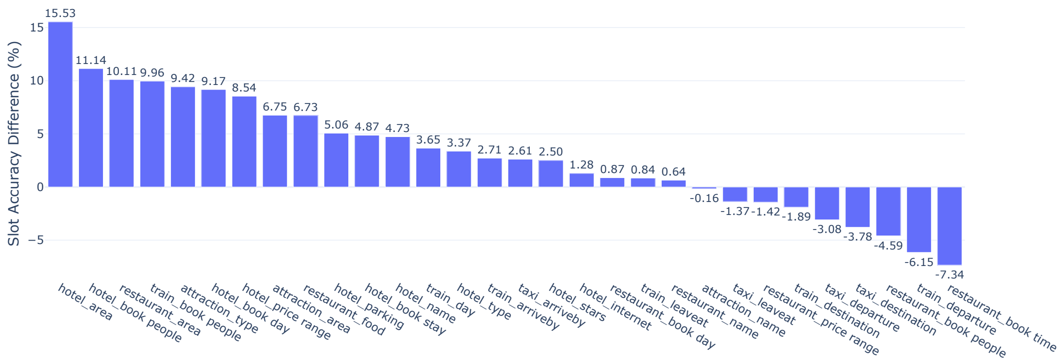

Figure 5: Slot accuracy difference between our model and (Li et al., 2021), a positive value indicates our model is better.

We show the slot accuracy for each unique slot in all five domains using 1% target domain training data with comparisons to generative baselines in Fig. 6.

B.4 Closest Tokens of Learned Prompt Tokens

Table 13 shows the full list of closest tokens for the most updated prompt tokens of each prompt segment in all five domains. We produce the closest tokens with the smallest cosine similarity between the learned prompt token embedding and frozen token embeddings of the PLM.

B.5 Variances of the Few-Shot Experimental Results

The variances of the experimental results reported in Table 1 (in the order of using 5, 10, 20 conversations and 1%, 5%, 10% of training data):

- Attraction: 0.27, 0.33, 0.35, 0.30, 0.22, 0.32
- Hotel: 0.52, 0.49, 0.55, 0.57, 0.61, 0.58
- Restaurant: 0.63, 0.72, 0.81, 0.79, 0.83, 0.80
- Taxi: 0.54, 0.61, 0.54, 0.48, 0.52, 0.69
- Train: 0.68, 0.72, 0.73, 0.52, 0.49, 0.55

B.6 Comparison with Frozen LM Version of Baselines

Table 7: Comparison with the frozen LM variation of the baseline. JGA (%) using 1% training data for each domain.

Model	Attr.	Hotel	Rest.	Taxi	Train
Li et al. (frozen LM)	29.16	14.81	15.14	47.56	35.77
Li et al.	42.18	24.04	30.70	58.26	45.32
Ours	47.28	33.01	34.40	60.33	52.05

Since there are many design choices to make to create a frozen LM version of the baselines (such as whether to add prefix prompt tokens, how to map tokens in the prompt segment to the underlying parameters etc), such variations would almost become new models. In our experiments, we show that our model outperforms existing models (optimizing all parameters) in low-resource settings, and we are confident that our model outperforms their frozen LM version with even larger gaps given the assumption that simply removing trainable parameters hurts the performance. We quantify such gaps by comparing the frozen and unfrozen version of the baseline Li et al. with our model in Table 7.

B.7 More about Error Analysis

Fig. 3 shows the error distribution in terms of the proportion of each error category, rather than the absolute error case counts. Though in Fig. 3, the omission error produced by our model takes the larger proportions in all five slot types compared with Li et al., our model actually makes fewer absolute omission errors than Li et al. in the "categorical" and "day" slot types, as shown in Table 8.

Table 8: Omission error counts divided by all testing instances (%) when training with 1% data.

Model	Open	Time	Number	Categorical	Day
Li et al.	**12.9**	**6.2**	**2.5**	7.2	2.8
Ours	16.4	8.3	4.8	**5.6**	**2.3**

We additionally show Slot Accuracy (SA) on the non-empty testing instances in Table 9. The result suggests that our model performs better than Li et al. in 4 out of 5 domains except for the "Taxi" domain, which is the most "none" dominated one.

Table 9: Slot Accuracy (%) on non-empty instances when training with 1% data.

Model	Attr.	Hotel	Rest.	Taxi	Train
Li et al.	55.34	66.37	72.17	**24.96**	84.70
Ours	**61.58**	**75.63**	**78.06**	19.74	**85.61**

Both observations indicate that even if omission error occupies more relative proportions of the error cases, our model is able to generate a reasonable amount of not-none values for non-empty slots compared with Li et al. in most domains.

C Details of Implementation and Experiments

C.1 Implementation Details

We apply different learning rate optimization for the parameters of each prompt segment. We use separate prompt embeddings for each prompt segment, meaning even if the same token appears in the prefix and question segments during initialization, it maps to different prompt embeddings for a larger optimization space. We use GPT2-medium with 1024 hidden states as our default model. We use the BPE tokenizer to convert the input sequence to tokens. We set the maximum sequence length to 1024. If the input sequence exceeds the maximum length, we cut the earlier part of the dialogue history while keeping the full other partition. Only the exact match between the generated sequence and the ground-truth slot value counts as a correct prediction. We use greedy decoding to generate the predicted sequence, and we stop the generation either when `<|endoftext|>` token is generated or the output length reaches 20. We choose the best epoch by monitoring JGA of the development set.

Our entire codebase is implemented in PyTorch.[6] The implementations of the transformer-based models are extended from the Huggingface[7] codebase (Wolf et al., 2020).

[6] https://pytorch.org/
[7] https://github.com/huggingface/transformers

C.2 Number of Task Prompt Tokens

We explore various values for the number of task prompt tokens used by the domain, slot and type prompt segments, and we show the hyper-parameters that lead to the best performance in Table 10. We observe that the domains with fewer unique slots (such as the attraction domain with just 3 unique slots) need much fewer prompt tokens than the domains with more unique slots (such as hotel and restaurant with 10 and 7 unique slots respectively). The more special prompt tokens needed, the more the parameter numbers are.

Table 10: Best prompt numbers for each domain

Model	Attr.	Hotel	Rest.	Taxi	Train
w/ reiteration	5	20	20	10	10
w/o reiteration	5	5	20	5	20

C.3 Experiment Details

We report the averaged result for three runs with different random seeds for each experiment. In the ablation study shown in Table 2, for lines 1-3, we directly remove the corresponding prompt segment from the input sequence; for lines 4-5, we keep the prefix and question text in the input sequence but use token embeddings rather than prompt embeddings to get initial token representation. In the prompt token semantic analysis in Table 3, we select the most changed prompt tokens by calculating the L2 norm of the difference of the learned and initialized prompt token embeddings.

All the models in this work are trained on a single Nvidia A6000 GPU on a Ubuntu 20.04.2 operating system. We show the hyperparameter search range and best hyperparameter setting in Table 11.

Table 11: Hyperparameter search range and the best setting.

Hyperparameter	Search Range	Best
Number of task prompt tokens	1, 2, 3, 5, 10, 15, 20, 25, 30	See Table 10
Prompt initialization	random, token embedding of task name	token embedding of task name
Batch size	1, 2, 3, 4	4
Learning rate	1e-2, 5e-3, 1e-3, 5e-4, 1e-4, 5e-5, 1e-5	1e-3
Decoding method	beam search, greedy	greedy
Surface form for empty slot	“none”, “not mentioned”, “”	“none”
Optimizer	Adam, Lamb (You et al., 2020)	Adam
Early stopping patience epochs		8
Learning rate scheduler		ReduceLROnPlateau with 5 patience epochs
Max epochs		100

C.4 Details of the Baseline Models

We produce the result of Li et al. and Lee et al. with our own reimplementation with our experimental setting and obtain the results of T5DST (Lin et al., 2021b) by running their codebase with our setting. We verify the correctness of our reproduction and we are able to reproduce the performance claimed in their papers under their settings. We report performance for TRADE and DSTQA from their papers. For Li et al., we use GPT2-medium as the backbone PLM and do not use DSTC8 for transfer learning as it would introduce additional data resources and make the comparison not fair. For T5DST, we use the best setting concluded by the authors that includes slot type information in the input sequence. We use T5-small with 60M parameters which has 6 encoder-decoder layers and the hidden size of 512 as the backbone PLM. For Lee et al., we use T5-small as the backbone, we include slot description from MultiWoZ 2.2, possible slot values from dialogue ontology and no domain description in the natural language augmented prompt.

D Limitations

There are several limitations to our work. Firstly, the proposed model is more sensitive to hyper-parameters such as the number of prompt tokens and learning rate than existing methods that fine-tune LM. Therefore, it would require additional parameter searching efforts to obtain the best performance. Secondly, our model is designed for and evaluated in English-only conversations, and applying our technique to other languages or code-switching scenarios might lead to performance decay. Finally, our experimental result shows that our proposed prompt tuning method works better than fine-tuning LM when there are fewer unique tasks to be optimized. Therefore, our method might not work well on a more diverse dataset.

E Ethics Statement

We do not see an immediate negative impact of the proposed method and do not see biased predictions made by our model. Our method is based on a pre-trained generative language model and trained on an open DST dataset, thus bias contained in the corpus for pre-training and the DST dataset might propagate to prediction outputs of our model. Human validation of the prediction results and their fairness needs to be conducted before our proposed model is used in production and other real-world applications. Our proposed model does not increase energy and carbon costs but will potentially reduce them due to its data and parameter efficiency.

Table 12: Question and description used in the input sequence for each slot

Domain	Slot	Question	Description
Attraction	area	In what area is the user looking for an attraction?	area to search for attractions
	name	What is the name of the attraction the user prefers?	name of the attraction
	type	What type of attraction does the user prefer?	type of the attraction
Hotel	area	In what area is the user looking for a hotel?	area or place of the hotel
	book day	The user is looking for a hotel starting what day of the week?	day of the hotel booking
	book people	How many people does the user need a hotel booking for?	number of people for the hotel booking
	book stay	How many days does the user prefer to stay at a hotel?	length of stay at the hotel
	internet	Does the user want internet in their hotel?	whether the hotel has internet
	name	What is the name of the hotel the user prefers?	name of the hotel
	parking	Does the user need the hotel to have parking?	whether the hotel has parking
	price range	What is the price range of the hotel the user prefers?	price budget of the hotel
	stars	The user prefers a hotel with what star rating?	star rating of the hotel
	type	What type of hotel does the user prefer?	what is the type of the hotel
Restaurant	area	In what area is the user looking for a restaurant?	area or place of the restaurant
	book day	The user is looking for a restaurant for what day of the week?	day of the restaurant booking
	book people	How many people does the user need a restaurant booking for?	how many people for the restaurant reservation
	book time	What time does the user want to book a restaurant?	time of the restaurant booking
	food	The user prefers a restaurant serving what type of food?	the cuisine of the restaurant you are looking for
	name	What is the name of the restaurant the user prefers?	name of the restaurant
	price range	What is the price range of the restaurant the user prefers?	price budget for the restaurant
Taxi	arrive by	What time does the user want to arrive by taxi?	arrival time of taxi
	departure	Where does the user want the taxi to pick them up?	departure location of taxi
	destination	Where does the user want to go by taxi?	destination of taxi
	leave at	What time does the user want the taxi to pick them up?	leaving time of taxi
Train	arrive by	What time does the user want to arrive by train?	arrival time of the train
	book people	How many people does the user need train bookings for?	how many train tickets you need
	day	What day does the user want to take the train?	day of the train
	departure	Where does the user want to leave from by train?	departure location of the train
	destination	Where does the user want to go by train?	destination of the train
	leave at	What time does the user want the train to leave?	leaving time for the train

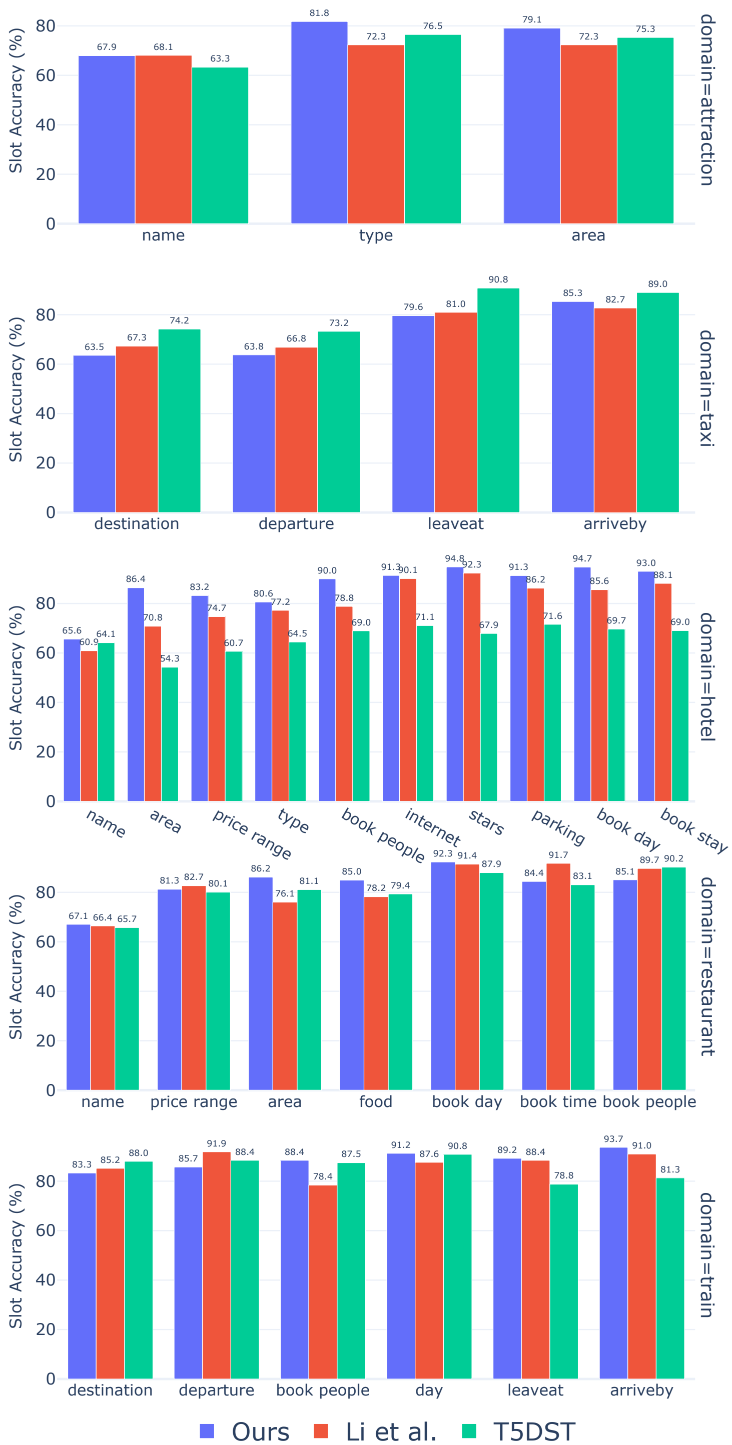

Figure 6: Slot accuracy for each slot across different domains

Table 13: Closest tokens for the learned prompt tokens

Domain	Prompt segment	Prompt token	Cloest tokens
Attraction	Domain	<domain_attraction_4> <domain_attraction_0>	raction; ractions; racted; ract; ractive att; Att; ATT; atts; atten
	Slot	<slot_name_2> <slot_type_4>	name; Name; names; NAME; named type; types; Type; style; TYPE
	Type	<type_open_3> <type_categorical_3>	open; Open; opened; opens; opening ateg; orical; orically; ategy; ategic
	Prefix	special site	special; Special; statistical; SPECIAL; remarkable site; sites; website; Site; webpage
	Question	Q attraction	answer; Answer; answered; answers; Q attraction; attractions; fascination; attractiveness; attracted
Hotel	Domain	<domain_hotel_11> <domain_hotel_19>	cogn; izoph; nostalg; contrad; Alas enment; Alas; ishy; ridic; minent
	Slot	<slot_parking_13> <slot_internet_17>	Pear; Aqua; Icar; Mermaid; Omega internet; Wi; Internet; WiFi; VPN
	Type	<type_number_14> <type_open_3>	regex; NUM; abulary; pmwiki; printf open; Dar; Ezek; Zur; Citiz
	Prefix	). yes	.).;).;].; .}; .</; .]; .); .'; }.; .) yes; Yes; YES; yeah; ye
	Question	hotel days	cannabis; sushi; Tinder; whiskey; booze days; hours; consequences; minutes; Days
Restaurant	Domain	<domain_restaurant_7> <domain_restaurant_16>	rest; Rest; urnal; restrial; restling rest; Rest; Rest; Text; Funds
	Slot	<slot_name_3> <slot_area_14>	name; Name; names; named; NAME area; Area; But; At; <|endoftext|>
	Type	<type_categorical_4> <type_day_15>	orical; brut; oric; ateg; day; Day; week; DAY; month
	Prefix	west booking	west; West; southwest; Southwest; northwest booking; reservation; insult; reverence; audition
	Question	name restaurant	exting; bookstore; describ; mascara; homepage Deadpool; Bitcoin; Veg; steak; Hollywood
Taxi	Domain	<domain_taxi_6> <domain_taxi_9>	i; a; o; I; in tax; Tax; taxes; Taxes; taxed
	Slot	<slot_destination_8> <slot_departure_4>	dest; Dest; destination; Destination ure; ures; URE; ured; uring
	Type	<type_time_0> <type_open_6>	time; Time; TIME; timer; year open; Open; opens; opening; closed
	Prefix	open rival	open; Open; opened; opens; OPEN rival; rivals; quickShip
	Question	taxi does	taxi; taxis; Taxi; cab; Uber does; is; doesn; has; isn
Train	Domain	<domain_train_8> <domain_train_7>	train; Train; genre; disciplinary; trained train; Train; trainers; trains; trained
	Slot	<slot_destination_8> <slot_arriveby_3>	dest; kosher; GMO; mill; JFK; okemon by; By; BY; While; from
	Type	<type_time_0> <type_day_6>	time; groupon; times; TIME; wasteful day; Day; week; month; year
	Prefix	dest location	dest; goto; inion; externalTo; Destination location; locations; geographic; geography; geographical
	Question	from train	graphs; ancestry; statistics; backstory; stats train; plane; railway; subway; highway