Show, Don't Tell: Demonstrations Outperform Descriptions for Schema-Guided Task-Oriented Dialogue

Raghav Gupta[*], **Harrison Lee**[*], **Jeffrey Zhao, Abhinav Rastogi, Yuan Cao, Yonghui Wu**
Google Research
{raghavgupta, harrisonlee}@google.com

Abstract

Building universal dialogue systems that operate across multiple domains/APIs and generalize to new ones with minimal overhead is a critical challenge. Recent works have leveraged natural language descriptions of schema elements to enable such systems; however, descriptions only indirectly convey schema semantics. In this work, we propose *Show, Don't Tell*, which prompts seq2seq models with a labeled example dialogue to *show* the semantics of schema elements rather than *tell* the model through descriptions. While requiring similar effort as generating descriptions, we show that using short examples as schema representations with large language models results in state-of-the-art performance on two popular dialogue state tracking benchmarks designed to measure zero-shot generalization - the Schema-Guided Dialogue dataset and the MultiWOZ leave-one-out benchmark.

1 Introduction

Task-oriented dialogue (TOD) systems need to support an ever-increasing variety of services. Since many service developers lack the resources to collect data and train models, zero and few-shot transfer to unseen services is critical to the democratization of dialogue agents.

Recent approaches to generalizable TOD systems primarily rely on combining two techniques: large language models like BERT (Devlin et al., 2019) and T5 (Raffel et al., 2020), and schema-guided modeling - i.e. using natural language descriptions of schema elements (intents and slots) as model inputs to enable transfer to unseen services (Rastogi et al., 2020a,b). Models combining the two currently hold state-of-the-art (SotA) results on dialogue state tracking (DST) (Heck et al., 2020; Lee et al., 2021a; Zhao et al., 2022).

However, description-based schema representations have some drawbacks. Writing precise natural language descriptions requires manual effort and can be difficult to write succinctly. Also, descriptions only provide indirect supervision about how to interact with a service compared to an example. Furthermore, Lee et al. (2021b) showed that schema-guided DST models are not robust to variations in schema descriptions, causing significant quality drops.

We propose using a single dialogue example with state annotations as an alternative to the description-based schema representation, similar to one-shot priming (Brown et al., 2020) - an approach we call *Show, Don't Tell (SDT)*. Through demonstration, we **show** models the schema semantics rather than **tell** them through natural language descriptions, as seen in Figure 1. SDT achieves SotA accuracy and generalization to new APIs across both the Schema-Guided Dataset (SGD) (Rastogi et al., 2020b) and MultiWOZ Leave-One-Out (Budzianowski et al., 2018; Lin et al., 2021b) benchmarks, while being more data-efficient and robust to schema variations.

2 Show, Don't Tell

Following SoTA models, we pose DST as a seq2seq task (Wu et al., 2019; Zhao et al., 2021a) and fine-tune T5 on DST datasets. The model input consists of a *prompt* to convey API semantics and *context* to represent the current dialogue instance. The *target* contains ground truth belief states corresponding to the context. We compare against two baselines:

- **T5-ind** (Lee et al., 2021a): Model input comprises *a single slot description* for the prompt, concatenated with the dialogue history as the context. The target is the value of the single slot in the dialogue state. Model inference is invoked once per slot - i.e. values for different slots are independently decoded.

- **T5-seq** (Zhao et al., 2022): Model input comprises the descriptions of *all slots* as the prompt,

[*]Equal contribution

T5-ind	SDT-ind
P_1 = amount: The amount of money to send or request P_2 = receiver: Name of the contact or account to make the transaction with ...	P_1^{ind} = [ex] [user] I need to transfer 125 dollars [slot] amount=125 dollars P_2^{ind} = [ex] [user] Make the transfer to Victoria. [slot] receiver=Victoria ...
T5-seq	**SDT-seq**
P = 0: The amount of money to send or request 1: Name of the contact or account to make the transaction with 2: Whether the transaction is private or not a) True b) False 3: The source of money used for making the payment a) credit card b) debit card c) app balance	P^{seq} = [ex] [user] I want to make a payment to Jerry for $82 from my mastercard [system] Confirming you want to pay Jerry $82 with your credit card yes? [user] Yes that's right, make the transaction private too [slot] amount=$82 receiver=Jerry private_visibility=a of a) True b) False payment_method=a of a) credit card b) debit card c) app balance

Figure 1: Illustration of all prompt formats for a payment service for both description-based and *Show, Don't Tell* models with independent (top) and sequential (bottom) decoding of dialogue state.

concatenated with the dialogue history as the context. The target is the sequence of slot-value pairs in the dialogue state - i.e. the dialogue state is decoded sequentially in a single pass.

We modify the prompt formats above to utilize demonstrations instead of descriptions as described below and illustrated in Figure 1.

- **SDT-ind**: A *prompt* P_i^{ind} comprises a single example utterance and the ground truth slot-value pair formatted as

$$P_i^{ind} = [ex]; u_i^{ind}; [slot]; sv_i$$

where u_i^{ind} is a user utterance where slot i is active/not null and sv_i is the slot-value pair. $[ex], [slot]$ are special delimiter tokens, and ; denotes concatenation.

- **SDT-seq**: A *prompt* P^{seq} comprises a single labeled dialogue formatted as:

$$P^{seq} = [ex]; u_1; ...; u_n; [slot]; sv_1; ...; sv_m$$

where u_j is an utterance, and other symbols are explained in the SDT-ind section above. In simple terms, the prompt is constructed by concatenating all utterances in an example dialogue followed by all slot-value pairs in the dialogue state.

In both the T5-* and SDT-* approaches, the context is the serialized dialogue history for the current dialogue instance. The final model input is formed by concatenating the prompt and the context strings, and the target string is the same as T5-*, containing only a single slot value for *-ind models and the entire turn's belief state for *-seq models.

For both T5-* and SDT-*, we enumerate the categorical slot values in multiple-choice format in the prompt and task models with decoding the multiple choice letter corresponding to the correct categorical value.

More details on prompt design and its impact on performance are provided in Appendix A.

Creating prompt examples: It is imperative that SDT prompts contain enough information to infer the semantics for all slots in a schema. For SDT-ind, we create individual utterances that showcase a single slot. For SDT-seq, we create example dialogues where all slots in the schema are used.

Multi-domain examples: It is not feasible to construct multi-domain demonstrations for every combination of domains. Thus, we stick to single-domain SDT prompts and create separate training instances for each domain present in a multi-domain dialogue turn; for inference, we run inference once for each domain and combine the results.

3 Experimental Setup

Datasets: We conduct experiments on two DST benchmarks: Schema-guided Dialogue (SGD) (Rastogi et al., 2020b) and MultiWOZ 2.1 (Budzianowski et al., 2018; Eric et al., 2020). For MultiWOZ, we evaluate on the leave-one-out setup (Wu et al., 2019; Lin et al., 2021a), where models are trained on all domains but one and evaluated on the holdout domain. Additionally, we apply the recommended TRADE pre-processing script[1] for fair comparison with other work. For both datasets, we created concise example dialogues modeled after

[1] https://github.com/budzianowski/multiwoz#dialog-state-tracking

dialogues observed in the datasets.

Implementation: We train SDT models by fine-tuning pretrained T5 1.1 checkpoints. For SDT-seq, we select one example dialogue for each service to create a prompt and use that prompt across all dialogue instances of that service, across training and evaluation. We do the same for SDT-ind but create one prompt per slot instead of per service. Unless otherwise noted, all T5-based models are based on T5-XXL (11B parameters). Appendices B and C contain more details on training and baselines respectively.

4 Results

4.1 SGD Results

Table 1 contains results on the SGD test set. SDT-seq achieves the highest JGA by +1.1%, outperforming the description-based T5-* models, particularly on unseen services. SDT-ind is comparable to its counterpart T5-ind and better than T5-seq.

Since SDT results vary with the choice of example dialogue provided in the prompt, we created 5 different versions of prompts for each service using different examples. We report the average JGA across the 5 versions and the 95% confidence intervals using the Student's-t distribution.

We hypothesize that the main advantage of SDT is that the schema semantics are conveyed via demonstration, which is more similar in form to the end task of state tracking and more informative than descriptions. On the other hand, natural language descriptions can be viewed as an intermediary that models must interpret in order to achieve the end goal of slot value prediction.

We see that SDT-seq outperforms SDT-ind and posit that this is because the full dialogue prompts in SDT-seq demonstrate more complex linguistic patterns (e.g. coreference resolution, long term dependencies) than the single utterance prompts of SDT-ind. On the other hand, we believe T5-seq does not outperform T5-ind because no additional information is conveyed to the model through concatenating independent descriptions. All-else-equal, decoding all slots in one pass is more challenging than decoding each slot independently.

We also experimented with using up to 5 example dialogues in each prompt of SDT-seq, but accuracy did not increase.

Model	All	Seen	Unseen
MRC+WD-DST*	86.5	92.4	84.6
T5-seq	86.4	**95.8**	83.3
T5-ind	87.7	95.3	85.2
SDT-ind	87.5±0.9	95.2±0.7	85.0±1.4
SDT-seq	**88.8±0.5**	**95.8±0.2**	**86.4±0.7**

Table 1: SDT achieves state-of-the-art JGA as evaluated on the SGD test set, performing especially well on unseen services. *Data augmentation/special rules applied.

Model	Attraction	Hotel	Restaurant	Taxi	Train	Avg
TRADE	20.1	14.2	12.6	59.2	22.4	25.7
SUMBT	22.6	19.8	16.5	59.5	22.5	28.2
TransferQA	31.3	22.7	26.3	61.9	36.7	35.8
T5-seq	**76.1**	28.6	69.8	**87.0**	60.4	64.4
SDT-seq	73.2	**34.2**	**71.9**	83.7	**68.4**	**66.3**

Table 2: SDT-seq outperforms T5-seq on the MultiWOZ 2.1 cross-domain (leave-one-out) benchmark. Results for TRADE, SUMBT, and TransferQA from Kumar et al. (2020), Campagna et al. (2020), and Lin et al. (2021a), respectively.

4.2 MultiWOZ Results

Table 2 summarizes results for the MultiWOZ 2.1 leave-one-out setup. SDT-seq outperforms T5-seq by +1.9% overall and in 3 of the 5 domains, achieving state-of-the-art performance.

4.3 Impact of Model Size

T5's XXL size (11B parameters) may be unsuitable in resource-constrained settings. To understand how the the impact of model size, we measure SDT's performance on SGD across multiple T5 sizes in Table 3. For base and large sizes, both SDT variations offer higher JGA than their description-based counterparts, possibly due to smaller T5 models being less capable of inferring unseen slots with just a description, whereas SDT models provide more direct supervision in contrast. Additionally, SDT-ind outperforms SDT-seq for both the smaller sizes, potentially due to SDT-seq's prediction task being more complex than that of SDT-ind.

4.4 Data Efficiency

To examine the data efficiency of SDT models, we also experiment with training SDT-seq with 0.16% (10-shot), 1%, and 10% of the SGD training data and evaluating on the entire test set. For 10-shot, we randomly sample 10 training dialogues from every service; for 1% and 10%, we sample uniformly across the entire dataset. SDT-seq demonstrates

Model	Base (250M)	Large (800M)	XXL (11B)
T5-seq	72.9	80.0	86.4
T5-ind	72.6	82.2	87.7
SDT-ind	**78.2±0.6**	**83.7±0.8**	87.5±0.9
SDT-seq	76.3±1.6	83.2±0.6	**88.8±0.5**

Table 3: SGD test set JGA across T5's Base, Large, and XXL sizes. SDT's advantage is especially prominent on smaller model sizes.

far higher data efficiency than T5-seq (Table 4), indicating that SDT is more suitable for bootstrapping dialogue systems with a limited budget for collecting training data.

Model	10-shot	1%	10%
T5-seq	51.0	79.4	83.0
SDT-seq	**70.7**	**84.5**	**87.4**

Table 4: Data efficiency experiments on the SGD test set. SDT-seq's example-based prompt approach is more suited to low resource settings than T5-seq's description-based prompts.

4.5 Robustness

Large LMs are often sensitive to the choice of prompt (Zhao et al., 2021b; Reynolds and McDonell, 2021). To this end, we evaluate SDT-seq on the SGD-X (Lee et al., 2021b) benchmark, comprising 5 variants with paraphrased slot names and descriptions for every schema (Appendix Figure 4). Note that SDT-seq only makes use of slot names, so variations in description have no effect on it.

Table 5 shows SDT-seq achieves the highest average JGA ($JGA_{v_{1-5}}$) and lowest schema sensitivity (SS_{JGA}), making it the most robust of the compared models. While the JGA decline indicates that SDT-seq is sensitive to how slot names are written, it is still more robust than a variant of T5-seq (Zhao et al., 2022) that only uses slot names.

Model	JGA_{Orig}	$JGA_{v_{1-5}}$	$Diff_{rel}$	SS_{JGA}
SGP-DST*	60.5	49.9	-17.5	51.9
T5-ind$_{base}$*	72.6	64.0	-11.9	40.4
T5-seq (name)★	79.7	73.0	**-8.4**	35.0
T5-seq	86.4	77.8	-10.0	27.0
SDT-seq	**88.8**	**81.2**	-8.6	**24.1**

Table 5: Robustness evaluation on the SGD-X test sets. *Results from Lee et al. (2021b). ★Result of using T5-seq with only slot names and no descriptions, from Zhao et al. (2022).

5 Discussion

5.1 Writing descriptions vs. demonstrations

The information provided to SDT is not identical to what is provided to typical schema-guided models, as SDT exchanges natural language descriptions for a demonstration of identifying slots in a dialogue. However, we argue that from the developer standpoint, creating a single example is similar in effort to writing descriptions, so we consider the methods comparable. Creating the SDT-seq prompts for all 45 services in SGD took an experienced annotator ∼2 hours, compared to ∼1.5 hours for generating all slot descriptions. SDT-ind prompts are even simpler to write because they relax the requirement for creating a coherent dialogue involving all slots.

Descriptions can sometimes be easier to generate than a succinct dialogue that covers all slots. However, given the performance gain, example-based prompts may be a better choice for many settings, especially for smaller model sizes and low resource settings where the gain over description-based prompts is more pronounced.

5.2 Descriptions plus demonstrations

We tried combining both descriptions and a demonstration in a single prompt to try to further improve performance. However, results showed that this did not improve upon using demonstrations alone (see Appendix Table A1 for details).

We hypothesize that demonstrations, along with slot names, already convey slot semantics sufficiently, rendering descriptions extraneous. However, given that using slot names alone underperforms using descriptions (Zhao et al., 2022), the improvement SDT exhibits over using descriptions does not result purely from the use of slot names.

5.3 Prompting vs. traditional finetuning

To understand the impact of using a single demonstration as a prompt vs. traditional finetuning, we finetune T5-seq an additional time on the same set of dialogues used in SDT-seq prompts; therefore it has access to both slot descriptions as well as a single demonstration for each service. In this case, T5-seq is provided strictly more information than SDT-seq. T5-seq with finetuning obtains a JGA of 87.7% on SGD, on par with T5-ind but still lower than SDT-seq, suggesting that, when scarce, dialogue examples are better used as prompts (Le Scao and Rush, 2021).

Example Dialogue	Predictions
1. Disambiguating similar slots [user] I need to find tickets to **Anaheim**, CA. [system] When would you like to travel, and where are you going to? [user] Traveling to **Sacramento** on the 4th.	T5-seq: *to=Sacramento, from=Anaheim* SDT-seq: *to=Anaheim, from=Sacramento*
2. Handling unseen slots [user] Can you please add an alarm called Grocery run.	T5-seq: *new_alarm_name=None* SDT-seq: *new_alarm_name=Grocery run*
3. Predicting categorical values not seen in prompt [user] I like Broadway shows and want to see one on Tuesday next week.	T5-seq: *event_type=theater* SDT-seq: *event_type=music*

Figure 2: Comparing common error patterns made by T5-seq vs. SDT-seq. Correct and incorrect predictions colored in red and blue, respectively.

Interestingly, finetuning on up to 5 dialogue examples per service did not improve performance after the first example (Appendix Figure 3).

5.4 Error analysis

Figure 2 compares some common error patterns made by T5-seq vs. SDT-seq. The patterns suggest that SDT's demonstrations are helpful when multiple slots in the same domain are similar to each other (#1 in Figure 2) and when slots dissimilar from those seen in training are introduced (#2). However, SDT can sometimes be limited by its prompt. For instance, in #3 it has only seen the "music" value for the *event_type* slot in the prompt, potentially resulting in under-predicting the categorical values not featured in the example dialogue (e.g. "theater").

6 Related Work

Prior approaches focused on framing DST as question answering (Ruan et al., 2020; Ma et al., 2019; Zhang et al., 2021). Many MultiWOZ cross-domain models leverage slot names/descriptions (Wu et al., 2019; Lee et al., 2019; Lin et al., 2021a).

Pretrained generative LLMs (Raffel et al., 2020; Brown et al., 2020) have enabled framing NLP tasks as seq2seq problems. Some DST papers (Zhao et al., 2021a; Feng et al., 2021) look at settings with no train-test discrepancy. Many studies explore the efficacy of task-specific prompts (Jiang et al., 2020; Liu et al., 2021). Madotto et al. (2020) and prime LMs with examples for dialogue tasks, but without finetuning. Wei et al. (2021) finetunes language models to teach them to use prompts to generalize across NLP tasks.

7 Conclusion

We study the use of demonstrations as LM prompts to convey the semantics of APIs in lieu of natural language descriptions for TOD. While taking similar effort to construct, demonstrations outperform description-based prompts in our experiments across DST datasets (SGD and MultiWOZ), model sizes, and training data sizes, while being more robust to changes in schemata. This work provides developers of TOD systems with more options for API representations to enable transfer to unseen services. In future work, we would like to explore this representation for other TOD tasks (e.g. dialogue management and response generation).

8 Ethical Considerations

We proposed a more efficient way of building TOD systems by leveraging demonstrations in place of descriptions, leading to increased accuracy with minimal/no data preparation overhead. We conduct our experiments on publicly-available TOD datasets in English, covering domains which are popular for building conversational agents. We hope our work leads to building more accurate TOD systems with similar or less overhead and encourages further research in the area.

References

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.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica Lam. 2020. Zero-shot transfer learning with synthesized data for multi-domain dialogue state tracking. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 122–132, Online. 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.

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

Yue Feng, Yang Wang, and Hang Li. 2021. A sequence-to-sequence approach to dialogue state tracking. 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 1714–1725, Online. Association for Computational Linguistics.

Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gasic. 2020. TripPy: A triple copy strategy for value independent neural dialog state tracking. In *Proceedings of the 21th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 35–44, 1st virtual meeting. Association for Computational Linguistics.

Zhengbao Jiang, Frank F. Xu, Jun Araki, and Graham Neubig. 2020. How can we know what language models know?

Norman P. Jouppi, Cliff Young, Nishant Patil, David Patterson, Gaurav Agrawal, Raminder Bajwa, Sarah Bates, Suresh Bhatia, Nan Boden, Al Borchers, Rick Boyle, and Pierre-luc et al. Cantin. 2017. In-datacenter performance analysis of a tensor processing unit. *SIGARCH Comput. Archit. News*, 45(2):1–12.

Adarsh Kumar, Peter Ku, Anuj Goyal, Angeliki Metallinou, and Dilek Hakkani-Tur. 2020. Ma-dst: Multi-attention-based scalable dialog state tracking. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8107–8114.

Teven Le Scao and Alexander Rush. 2021. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2627–2636, Online. Association for Computational Linguistics.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021a. 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.

Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. 2021b. Sgd-x: A benchmark for robust generalization in schema-guided dialogue systems.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-utterance matching for universal and scalable belief tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5478–5483, Florence, Italy. Association for Computational Linguistics.

Zhaojiang Lin, Bing Liu, Andrea Madotto, Seungwhan Moon, Paul Crook, Zhenpeng Zhou, Zhiguang Wang, Zhou Yu, Eunjoon Cho, Rajen Subba, and Pascale Fung. 2021a. Zero-shot dialogue state tracking via cross-task transfer.

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.

Xiao Liu, Yanan Zheng, Zhengxiao Du, Ming Ding, Yujie Qian, Zhilin Yang, and Jie Tang. 2021. Gpt understands, too.

Yue Ma, Zengfeng Zeng, Dawei Zhu, Xuan Li, Yiying Yang, Xiaoyuan Yao, Kaijie Zhou, and Jianping Shen. 2019. An end-to-end dialogue state tracking

system with machine reading comprehension and wide & deep classification.

Andrea Madotto, Zihan Liu, Zhaojiang Lin, and Pascale Fung. 2020. Language models as few-shot learner for task-oriented dialogue systems. *CoRR*, abs/2008.06239.

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.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020a. Schema-guided dialogue state tracking task at dstc8.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):8689–8696.

Laria Reynolds and Kyle McDonell. 2021. Prompt programming for large language models: Beyond the few-shot paradigm.

Yu-Ping Ruan, Zhen-Hua Ling, Jia-Chen Gu, and Quan Liu. 2020. Fine-tuning bert for schema-guided zero-shot dialogue state tracking.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. 2021. Finetuned language models are zero-shot learners.

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.

Yang Zhang, Vahid Noroozi, Evelina Bakhturina, and Boris Ginsburg. 2021. Sgd-qa: Fast schema-guided dialogue state tracking for unseen services.

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.

Jeffrey Zhao, Mahdis Mahdieh, Ye Zhang, Yuan Cao, and Yonghui Wu. 2021a. 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.

Tony Z. Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. 2021b. Calibrate before use: Improving few-shot performance of language models.

A Prompt Design

We experimented with various formats for the SDT prompt before arriving at the final format. Below, we list alternative designs that we tried and their impact on JGA, as evaluated on the SGD test set.

A.1 Categorical value strings vs. multiple choice answers

We found that JGA dropped -2% when we tasked the model with decoding categorical values instead of multiple choice answers - e.g. `payment_method=debit card` instead of `payment_method=b` (where `b` is linked to the value `debit card` in the prompt as described in Section 2). When tasking the model to decode categorical values, it would often decode related yet invalid values, which we counted as false in our evaluation. For example, instead of `debit card`, the model might decode `bank balance`.

A.2 Slot IDs vs. slot names

When we delexicalized slot names with slot IDs, JGA dropped -5%. One downside of this approach is that the model lost access to valuable semantic information conveyed by the slot name. Another downside is that the model could not distinguish two slots that had the same value in the prompt. For example, if the prompt was "I would like a pet-friendly hotel room with wifi" and the corresponding slots were `1=True` (has_wifi) and `2=True` (pets_allowed), it is ambiguous which ID refers to which slot.

The potential upside of using slot IDs was to remove dependence on the choice of slot name, but this did not succeed for the reasons above.

A.3 Decoding active slots vs. all slots

We experimented with training the model to only decode active slots rather than all slots with `none` values when they were inactive. JGA dropped -0.4%, which we hypothesized might be a result of greater dissimilarity between the slot-value string in the prompt (which contained all slots by construction) and the target, which only contained a subset of slots.

A.4 In-line annotations vs. dialogue+slots concatenated

We hypothesized that bringing the slot annotation in the prompt closer to where it was mentioned

in the dialogue might help the model better understand the slot's semantic meaning. We changed the format as follows:

- Original: `[ex] [user] I would like a pet-friendly hotel room with wifi [system] I found ...` **`[slot] has_wifi=True`**
- In-line: `[ex] [user] I would like a pet-friendly hotel room with wifi` **`[has_wifi=True]`** `[system] I found ...`

However, this decreased JGA by more than -20%. We hypothesized that this was likely due to a mismatch between the prompt's annotations and the target string format, which we did not change.

B SDT Model Details

We used the publicly available T5 checkpoints[2]. For all experiments, we used a sequence length of 2048, 10% dropout and a batch size of 16. We used a constant learning rate of $1e-3$ or $1e-4$. All models were trained for 50k steps or until convergence, and each experiment was conducted on either 64 or 128 TPU v3 chips (Jouppi et al., 2017).

C Baseline Models

For SGD, we compare against SGP-DST (Ruan et al., 2020), MRC+WD-DST (Ma et al., 2019), T5-seq (Zhao et al., 2022) and T5-ind (Lee et al., 2021a).

For MultiWOZ, we compare against TRADE (Wu et al., 2019), SUMBT (Lee et al., 2019), TransferQA (Lin et al., 2021a), and T5-seq. Transfer QA is based on T5-large.

Model	All	Seen	Unseen
SDT-seq + desc	88.6±0.9	95.7±0.5	86.2±1.0
SDT-seq	**88.8±0.5**	**95.8±0.2**	**86.4±0.7**

Table A1: We experiment with prompting using both descriptions and demonstrations (SDT-seq + desc) vs. demonstrations-only (SDT-seq) and find that adding descriptions does not improve performance.

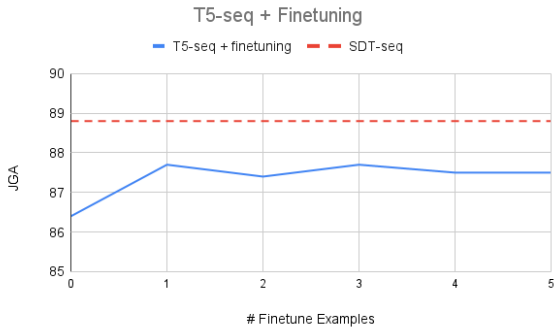

Figure 3: Results of secondarily finetuning T5-seq with dialogues, to help understand whether prompting or finetuning is more effective. The examples used for finetuning are derived from the set of dialogues used as prompts across the 5 trials of SDT-seq. From this, we observe that prompting with a single dialogue demonstration outperforms few-shot finetuning.

[2] https://github.com/google-research/text-to-text-transfer-transformer/blob/main/released_checkpoints.md

Original	V1	V5
service_name: "Payment" description: "The fast, simple way to pay in apps, on the web, and in millions of stores"	service_name: "Payment" description: "Best way to pay online or in-person"	service_name: "Payment" description: "Money transfers and payment requests made easy"
name: "amount" description: "The amount of money to send or request"	name: "amt" description: "Amount sent or requested"	name: "amount_to_transfer" description: "Cash amount to transfer or ask for"
name: "receiver" description: "Name of the contact or account to make the transaction with"	name: "recipient_info" description: "Name of person to receive payment or request"	name: "contact_name_or_account_name" description: "Payment will be sent to or requested from this person/entity"
name: "private_visibility" description: "Whether the transaction is private or not"	name: "visibility" description: "Boolean flag indicating if the transaction is private or not"	name: "private_transaction_yes_or_no" description: "Hidden transaction yes/no?"
name: "payment_method" description: "The source of money used for making the payment"	name: "payment_source" description: "Source of money for transfer"	name: "money_withdrawal_source" description: "What is being used to pay, either app balance or debit/credit card"
name: "RequestPayment" description: "Request payment from someone"	name: "RequestAPayment" description: "Request money from another user"	name: "TransferRequest" description: "Ask for a money transfer from a contact"
name: "MakePayment" description: "Send money to your friends"	name: "SendPayment" description: "Send cash to friends and others"	name: "TransferMoney" description: "Make a payment to an account"

Figure 4: The original schema for a Payment service alongside its closest (v_1) and farthest (v_5) SGD-X variants, as measured by linguistic distance functions. For the SGD-X benchmark, models are trained on the original SGD dataset and evaluated on the test set, where the original test set schemas are replaced by SGD-X variant schemas.