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

Anonymous ACL submission

Abstract

Building universal dialogue systems that can seamlessly operate across multiple domains/APIs and can generalize to new ones with minimal supervision and low maintenance is a critical challenge. Recent works have leveraged natural language descriptions for schema elements to build such systems. However, descriptions only provide indirect supervision for downstream tasks, while still requiring effort to construct. In this work, we propose *Show, Don't Tell*, which uses a short labeled example dialogue to *show* the semantics of a schema rather than *tell*ing the model about the schema elements via descriptions. While requiring similar effort from service developers, we show that using short examples as schema representations with large language models results in stronger performance and better generalization on two popular dialogue state tracking benchmarks: the Schema-Guided Dialogue (SGD) dataset and the MultiWoZ leave-one-out benchmark.

1 Introduction

With the widespread adoption of task-oriented dialogue (TOD) systems, these need to support an ever-increasing variety of services/APIs. Since many service developers lack the resources to collect labeled data or the requisite ML expertise, zero/few-shot transfer to unseen services becomes critical to the democratization of dialogue agents.

New approaches to TOD that can generalize to new services mainly 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 inference on unseen services (Rastogi et al., 2020a,b). Models combining the two currently show state-of-the-art results on dialogue state tracking (DST) (Heck et al., 2020; Lee et al., 2021a; Anon, 2021).

However, description-based schema representations have drawbacks: precise natural language descriptions still take manual effort and can be tricky to write, while only constituting indirect supervision for unseen services compared to an example dialogue. Furthermore, Lee et al. (2021b) showed that state-of-the-art schema-guided DST models may not be robust to variation in schema descriptions, causing significant accuracy drops.

Alternatively, we propose using a single dialogue example (with final state annotations) in place of the service schema representation, similar to one-shot priming (Brown et al., 2020). Rather than **tell**ing the model about schema element semantics in natural language, we aim to **show** the schema through a demonstration, as in Figure 1. Our approach, *"Show, Don't Tell (SDT),"* when applied to two SotA DST models, offers consistently superior accuracy and generalizes better to new APIs across both the 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), where the input language model (in our case, T5) is finetuned on the training set for a DST dataset. During finetuning and evaluation, the model input consists of a *prompt* and *context*, and the *target* contains ground truth belief states. We consider two models/prompt formats as our baselines:

- **T5-ind** (Lee et al., 2021a): Model input comprises of the dialogue history as context concatenated with *one slot description* as the prompt. The target is the value of that slot in the dialogue state. Inference must be done per slot i.e. values for different slots are independently decoded.
- **T5-seq** (Anon, 2021): Model input comprises

T5-ind
amount: The amount of money to send or request
receiver: Name of the contact or account to make the transaction with
...

SDT-ind
[ex] [user] I need to transfer 125 dollars [slot] amount=125 dollars
[ex] [user] Make the transfer to Victoria. [slot] receiver=Victoria

T5-seq
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

SDT-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 description-based as well as *Show, Don't Tell* models with a) independent (top) and b) sequential (bottom) decoding of dialogue state.

the descriptions of *all slots* as the prompt, followed by the dialogue history as the context. The target is the sequence of slot-value pairs in the dialogue state. In other words, the dialogue state is decoded sequentially in a single pass.

We modify the above prompt formats to include demonstrations instead of descriptions as follows. The new example-based prompt formats are described below and illustrated in Figure 1.

- **SDT-ind**: A *prompt* $\mathtt{P}_\mathtt{i}^{\mathtt{ind}}$ comprises a single labeled slot value pair for slot $\mathtt{i}$ formatted as

$$\mathtt{P}_\mathtt{i}^{\mathtt{ind}} = [\mathtt{ex}]; \mathtt{d}_\mathtt{i}^{\mathtt{ind}}; [\mathtt{slot}]; \mathtt{sv}_\mathtt{i}$$

where $\mathtt{d}_\mathtt{i}^{\mathtt{ind}}$ is a single user utterance indicating a value for slot i, and $\mathtt{sv}_\mathtt{i}$ is the slot value pair. $[\mathtt{ex}]$, $[\mathtt{slot}]$ are special delimiter tokens.

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

$$\mathtt{P}^{\mathtt{seq}} = [\mathtt{ex}]; \mathtt{d}_1; ...; \mathtt{d}_\mathtt{n}; [\mathtt{slot}]; \mathtt{sv}_1; ...; \mathtt{sv}_\mathtt{m}$$

i.e. the prompt is constructed by concatenating all utterances in the example dialogue followed by all slot-value pairs in the final dialogue state.

For all prompt formats (T5-* and SDT-*), we format the values for categorical slots (taking one of a fixed set of values) as a multiple-choice question.

The *context* in both prompt formats is a concatenation of the dialogue history for the current training example. The final model input is formed by concatenating the prompt and the context strings. The *target* string is unchanged, containing only the value for the specific slot for independent decoding and the turn belief state for sequential decoding. More details on the prompt design and its impact on performance are provided in Appendix C.

Formulating prompt examples: Given neither SDT prompt format contains slot descriptions, it is imperative that the prompt(s) contain enough semantic information to infer values for all slots in the schema. This is easy for SDT-ind, which uses a separate prompt for each slot. However, for SDT-seq, we ensure that the chosen example dialogue contains annotations for all slots in that schema.

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., 2019). For MultiWOZ, we evaluate on the cross-domain transfer setup from Wu et al. (2019); Lin et al. (2021a), where models are trained on all domains but one and evaluated on the holdout domain. For SGD, we created prompt dialogues manually, with 5.9 turns on average, compared to 15.3 average turns in the SGD single-domain dataset. For MultiWoZ, we selected a short dialogue containing all slots from each holdout domain's training set for the prompt.

Implementation: We train SDT models by fine-tuning pretrained T5 1.1 checkpoints of various sizes. For both datasets, we select one example prompt per service schema (for SDT-seq) or slot (for SDT-ind), and use the same prompt for all examples for that service/slot across training and evaluation. Unless otherwise noted, all T5-based models (T5/SDT-seq/ind) are finetuned on T5-XXL (11B parameters). Appendices A and B have more details on training and baselines respectively.

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±0.6	95.1±0.5	85.0±0.9
SDT-seq	**88.8±0.5**	**95.8±0.2**	**86.4±0.7**

Table 1: SGD test set JGA for SDT versus other approaches. *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	**76.4**	26.1	**74.9**	85.9	64.6	65.6
SDT-seq	75.0	**32.0**	73.1	**86.6**	**77.5**	**68.8**

Table 2: Cross-domain (leave-one-out) JGA on MultiWOZ 2.1. Results for TRADE, SUMBT, and TransferQA from (Kumar et al., 2020), (Campagna et al., 2020), and (Lin et al., 2021a), respectively.

4 Results

4.1 Results on SGD

Table 1 contains results on the SGD test set. Since SDT results may depend on the choice of example turn/dialogue provided in the prompt, 5 different versions of prompts are created for each service using different examples. The reported results are obtained by averaging the JGA across these versions. SDT-seq achieves the highest JGA, showing major gains, particularly over unseen services, over its description-based counterpart T5-seq and the next-best model T5-ind. SDT-ind is comparable to its counterpart T5-ind, and better than T5-seq.

Based on these results, a single dialogue example appears more effective than using natural language descriptions. By its construction, the SDT-ind prompt format is unable to model phenomena such as coreference, a limitation not faced by SDT-seq which can jointly model all slots in a service.

Further finetuning T5-seq: To evaluate T5-seq in a scenario where it can access the dialogue examples used for SDT-seq prompts, We try further finetuning T5-seq on this exact set of dialogue examples. This model, therefore, gets slot descriptions as well as the demonstrations to finetune on. The model obtains a JGA of 87.7% on SGD, level with T5-ind but still lower than SDT-seq, indicating dialogue examples are better used as prompts (Le Scao and Rush, 2021). Interestingly, finetuning on more than one dialogue example does not help.

4.2 MultiWOZ Results

Table 2 summarizes results for the MultiWOZ 2.1 leave-one-out transfer setup. Comparing T5-seq and SDT-seq, both finetuned over T5-XXL, the latter achieves state-of-the-art results on 3 of 5 domains and overall for this task, and the former performs best for the remaining 2 domains.

4.3 Impact of Model Size

T5-XXL may be too large/slow for a number of settings, so we look SDT's performance on SGD across more T5 model sizes in Table 3. For the base and large model sizes, SDT variants offer consistently higher JGA than their description-based counterparts. SDT-ind fares better than SDT-seq, possibly due to smaller T5 models being less capable of inferring unseen slots with just a description.

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±0.4**	**83.7±0.6**	87.5±0.6
SDT-seq	76.3±1.1	83.2±0.4	**88.8±0.5**

Table 3: SGD test JGA across T5 model sizes.

4.4 Data Efficiency

To examine the data efficiency of SDT models, we train SDT-seq in a low-resource setting 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 dialogues from every service; for 1% and 10%, we sample uniformly from the full dataset. Results from Table 4 demonstrate far higher training data efficiency for SDT-seq.

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 SGD test set.

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) dataset, which includes 5 variant schemas with paraphrased slot names and descriptions. Table 5 shows SDT-seq achieves the highest average JGA ($JGA_{v_{1-5}}$) and lowest schema sensitivity (SS_{JGA}), indicating it

Example Dialogue Segment	Error
1. T5-seq confused between similar slots I need to find train tickets to Anaheim, CA. When would you like to travel, and where are you going to? Traveling to Sacramento on the 4th.	T5-seq swaps values for slots *from* and *to*
2. Slot values appearing as in context Can you please add an alarm called Grocery run.	T5-seq misses slot *new_alarm_name*
3. Categorical values not seen in context I like Broadway shows and want to see one on Tuesday next week.	SDT-seq misses *event_type=theater*

Figure 2: Examples of common error patterns made by SDT-seq compared to T5-seq.

is the most robust of the compared models. Note, however, that the JGA drop indicates SDT-seq is still sensitive to slot name variations.

Model	JGA_{Orig}	$\text{JGA}_{v_{1-5}}$	Diff_{rel}	SS_{JGA}
SGP-DST*	60.5	49.9	-17.6	51.9
T5-ind_{base}*	72.6	64.0	-11.9	40.4
T5-seq	86.4	77.8	-8.6	27.0
SDT-seq	**88.8**	**81.2**	**-7.6**	**24.1**

Table 5: Robustness evaluation on the SGD-X test sets. *Results from Lee et al. (2021b).

5 Discussion

5.1 Writing descriptions vs. demonstrations

We note that the information provided to SDT is not identical to what is provided to usual schema-guided models, as SDT trades out natural language descriptions in exchange for a demonstration of how to identify slots in a dialogue. However, we argue that from a developer's point of view, creating a single example is a similar amount of effort as writing descriptions, so we consider the methods to be comparable. For creating the SDT-seq prompts for each service in SGD, an experienced annotator took ~2 hours, compared to ~90 minutes for generating descriptions for all slots in all services. SDT-ind prompts are arguably even simpler to write.

Descriptions, however, have their advantages: they are agnostic to the model architecture and writing them does not require knowledge of dialogue systems, which SDT-seq prompts does. Given the performance gain, however, example-based prompts may be a better choice for many settings, especially for smaller model sizes.

5.2 Error analysis

Figure 2 contains some common error patterns in predictions from T5-seq and SDT-seq. These indicate that SDT benefits from having a better understanding of the context around unseen slots, or when slot descriptions are too similar to one another (#1). However, SDT can be limited by its prompt: for instance, in #3 it has only seen context for the other categorical value for slot *event_type* .

6 Related Work

Prior approaches focused on framing DST as question answering (Ruan et al., 2020; Ma et al., 2020; 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 enable 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) prime LMs with examples for dialogue tasks, but without finetuning. Wei et al. (2021) fine-tune language models to understand prompts for a different task.

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. Even though they take similar effort to construct, they 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 for. We are hopeful our work leads to building more accurate TOD systems with similar or less overhead, and encourages further research in the area.

References

Anon. 2021. Description-driven task-oriented dialog state tracking. *ACL Rolling Review, November 2021*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared 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 M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher 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.

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*.

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.

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, Shuyag Gao, and Dilek Hakkani-Tur. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv preprint arXiv:1907.01669*.

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. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8107–8114.

Teven Le Scao and Alexander M 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.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021a. Dialogue state tracking with a language model using schema-driven prompting. *arXiv preprint arXiv:2109.07506*.

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 Conference of the Association for Computational Linguistics*, pages 5478–5483.

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

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. 2020. 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. *arXiv preprint arXiv: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. *arXiv preprint arXiv:2002.01359*.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020b. 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.

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. *arXiv preprint arXiv:2002.00181*.

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, 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.

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

A SDT Model Details

All T5 checkpoints used are available publicly[1]. For all experiments, we use a sequence length of 2048, dropout of 10% 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).

B Baseline Models

For SGD, we compare against SGP-DST (Ruan et al., 2020), MRC+WD-DST (Ma et al., 2020), T5-seq (Anon, 2021) 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, and T5-ind and T5-seq are based on T5-XXL in this paper unless otherwise noted.

C 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.

C.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). We found that 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

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

of `debit card`, the model might decode `bank balance`.

C.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.

C.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 could 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.

C.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: `[example] [user] I would like a pet-friendly hotel room with wifi [system] I found ...` **`[slot] has_wifi=True`**
- In-line: `[example] [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 remained the same.