More Robust Schema-Guided Dialogue State Tracking via Tree-Based Paraphrase Ranking

Alexandru Coca[†], **Bo-Hsiang Tseng**[‡], **Weizhe Lin**[†], **Bill Byrne**[†]
[†]Department of Engineering, University of Cambridge, United Kingdom
[‡]Apple
{ac2123, wl356, wjb31}@cam.ac.uk
bohsiang_tseng@apple.com

Abstract

The schema-guided paradigm overcomes scalability issues inherent in building task-oriented dialogue (TOD) agents with static ontologies. Instead of operating on dialogue context alone, agents have access to hierarchical schemas containing task-relevant natural language descriptions. Fine-tuned language models excel at schema-guided dialogue state tracking (DST) but are sensitive to the writing style of the schemas. We explore methods for improving the robustness of DST models. We propose a framework[1] for generating synthetic schemas which uses tree-based ranking to jointly optimise lexical diversity and semantic faithfulness. The generalisation of strong baselines is improved when augmenting their training data with prompts generated by our framework, as demonstrated by marked improvements in average joint goal accuracy (JGA) and schema sensitivity (SS) on the SGD-X benchmark.

1 Introduction

DST is concerned with tracking user goals in task-oriented conversations. The goals are represented as key-value pair sequences, with the keys known as *slots* (e.g. hotel name). Pre-trained language models (PLMs) (Devlin et al., 2019; Raffel et al., 2020) have helped shift focus from systems that can only track slots drawn from a database or *domain ontology* (Henderson et al., 2014) to models that do not require re-training to parse goals in new domains. The Schema-Guided Dialogue (SGD) dataset (Rastogi et al., 2020) facilitates this shift with a large-scale set of conversations grounded in 45 *service APIs* or *schemas* that describe the domains, slots and user intents that annotate the conversations (Appendix A). Test set dialogues are grounded in 6 schemas *seen* during training and 15 *unseen* ones.

Neural models perform impressively on the difficult schema-guided DST task (Rastogi et al., 2020), but Lee et al. (2022) show that the uniformity of the descriptive language of the schemas facilitates this. They create the SGD-X benchmark to evaluate robust zero-shot generalisation of DST models. This is achieved by grounding the SGD test set conversations in five *schema variants* increasingly dissimilar to the SGD schemata[2]. To perform well, a DST model should correctly track the state of a dialogue when conditioned, in turn, on prompts constructed from the five variants.

We show how to improve DST robustness by introducing controlled variability in the data. We contribute to robust DST research by (1) a flexible framework for generating and ranking diverse outputs of a paraphrase model based on a tree-clustering algorithm designed to control lexical diversity and semantic similarity; (2) combine state-of-the-art paraphrase models and language generation metrics to generate increasingly diverse schemata paraphrases; (3) show that augmenting the training dataset with these schemata improves the robustness and generalisation performance of strong DST baselines.

2 Related Work

Input variety, data scarcity and domain shifts affect the robustness of DST models. Liu et al. (2021) investigate the former. They employ word-level data augmentation (DA) (Wei and Zou, 2019), turn paraphrasing and speech disfluency modeling to approximate their field performance. Turn and dialogue generation are effective in low-resource settings (Campagna et al., 2020; Hou et al., 2018) but are very difficult to scale to new domains and are not effective in the high-resource setting we consider (Campagna et al., 2020; Mohapatra et al., 2021). This also applies to word- and sentence-level meth-

[1]Code will be released here: `https://bit.ly/3WYB7Fl`

[2]Variants are ordered according to their lexical similarity to the SGD schemas. The v1 variant is the most similar whereas v5 is the most dissimilar. See Appendix A for details and examples and the schemata here: https://bit.ly/3Ev0KrV.

ods (Quan and Xiong, 2019; Louvan and Magnini, 2020). Lee et al. (2022) find word-order changes and deletions to be ineffective in the high-resource, schema-guided setting we consider.

Schema-guided DST tackles both data scarcity and novel domains by using API definitions to prompt PLMs (Zhao et al., 2022). Yet Lee et al. (2022) demonstrate the lack of robustness of schema-guided DST models to prompt styles and vocabulary, creating a new research direction. They show that augmenting the training data with synthetic prompts obtained via backtranslation significantly improves models' ability to track states under meaning-preserving prompt transformations. Backtranslation is also applied to improve DST robustness to linguistic variation inherent in user communication (Ma et al., 2019; Einolghozati et al., 2019), which is orthogonal to the prompt style and vocabulary robustness setting we consider. Reinforcement learning has also been applied (Yin et al., 2020), but works only in the very constrained single-domain, ontology-driven setting. Other TOD-relevant DA approaches apply to policy learning (Gritta et al., 2021) and response-generation (Gao et al., 2020; Zhang et al., 2020b).

Addressing the dearth of augmentation methods designed to ensure prompt robustness of schema-guided DST models, we propose to generate schemas by ranking large paraphrase candidate lists with learned metrics in a tree ranking scheme.

3 Tree-Based Paraphrase Ranking

Tree construction A large pool of schema candidates is created by generating paraphrases given grids of generation parameters (eg temperature, number of beams). The set is filtered to address generation failures (eg toxic and hallucinated words). We optionally filter candidates with an entailment model to increase semantic faithfulness (Narayan et al., 2022) (see Appendix B.1).

The tree constructor (Algorithm 1) takes as input an object (`Node`) that stores a metric value, `val`, and the candidate paraphrases which are split at that node, `sents`. A list of metrics to be computed between each candidate and the input is provided by the user. This enables our framework to build arbitrary-depth trees with custom user metrics. Each unique list of metric values describing the distance between the input and a candidate generates a path in the tree (lines 5-13). The n-ary tree constructed in this way has the property that level-order traversal of the first level can yield diverse candidates with respect to the metric it encodes. In practice, the metrics measure lexical and semantic distances between their inputs.

Algorithm 1: Tree building

```
1:  def build_tree(root: Node, inp: str,
      cands: list[str], metrics: list[Callable]):
        Data: root, inp input , cands input
              paraphrases, metrics objects to
              eval. dist. between input & cand.
        Result: tree splitting cands according
                to metrics
2:      curr ← root ;
3:      for c in cands:
4:          curr ← root ;
5:          for m in metrics:
6:              m_val = m(inp, c) ;
7:              next ← get_child
                 (curr.children, m_val) ;
8:              if next is NULL:
9:                  next ← Node (val=m_val,
                     sents=[c]) ;
10:                 curr.children.add(next)
11:             else:
12:                 next.sents.add(c) ;
13:                 curr ← next
14:     return root
```

Ranking Our ranker input is the tree and a list of decision functions, with elements corresponding to each level in the tree, `f_dec`. Without loss of generality, we assume that the first level encodes a metric with respect to which the user wishes to maximise diversity (eg lexical distance). As shown in Figure 1, our algorithm traverses breadth-first the level for which diversity is to be maximised. Each subtree returned in the traversal is traversed depth-first, guided by the decision functions. For example, in Figure 1 we show that the node $B = 0.77$ is selected by applying the max decision function to the children of $J = 66$, and that applying min to the children of $B = 0.77$ selects the leaf $S = 77$. See Algorithm 3 (Appendix B) for details.

4 Experiments

4.1 Schema generation

Our paraphrase model is PegasusParaphrase[3], a fine-tuned Pegasus model (Zhang et al., 2020a).

[3]Available at https://bit.ly/3vgY7EZ.

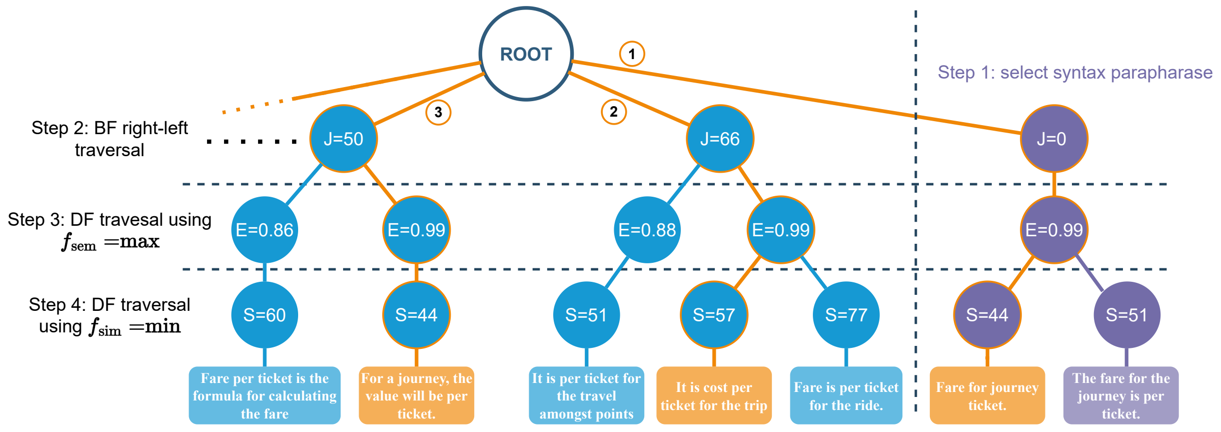

ROOT
1
2
3
Step 1: select syntax parapharase
Step 2: BF right-left traversal
Step 3: DF travesal using $f_{\text{sem}} = \max$
Step 4: DF traversal using $f_{\text{sim}} = \min$
J=50
J=66
J=0
E=0.86
E=0.99
E=0.88
E=0.99
E=0.99
S=60
S=44
S=51
S=57
S=77
S=44
S=51
Fare per ticket is the formula for calculating the fare
For a journey, the value will be per ticket.
It is per ticket for the travel amongst points
It is cost per ticket for the trip
Fare is per ticket for the ride.
Fare for journey ticket.
The fare for the journey is per ticket.

Figure 1: Ranking paraphrases of *Fare per ticket for journey* using a tree. Top level node split is by Jaccard distance (J), middle nodes split by entailment score (E) and leave nodes store string similarities (S). Here $f_{\text{dec}} = [\text{None}, \max, \min]$. By using J we guarantee that candidates with $J = 0$ are syntactic paraphrases if S is constrained. Orange leaves show top ranked candidates. Numbers on paths show ranking order.

Using 10 settings for the number of beams and temperature, we generate 500 candidates for each input. For efficiency purposes, these are filtered heuristically (Appendix B). We construct a depth $d = 3$ tree which splits the candidates by Jaccard distance J, entailment E, and string similarity S. That is, the input to our tree constructor (Algorithm 2) is metrics $= [J, E, S]$. Here entailment is computed using BART (Lewis et al., 2020) as described in Appendix B. We prune nodes with $J > 0.75$ to limit hallucination.

We select $k = 5$ lexically-diverse paraphrases that maximise entailment given the constraint that the returned candidates should be lexically diverse. First, we select a syntactic paraphrase by traversing the subtree rooted at $J = 0$ and minimising S. The remainder of the candidates are selected by constraining the bread-first traversal of the first level, which encodes lexical distance, to return the nodes sorted from high to low. This procedure is depicted schematically in Figure 1. We sort the ranked candidate lists for each description according to the Jaccard distance between them and the SGD descriptions. Hence we obtain $k = 5$ synthetic schema variants, with $v1$ being the most similar to the SGD schema and $v5$ the most dissimilar. We refer to this scheme as `Pegasus + BART`.

4.2 State tracking data augmentation

4.2.1 Baseline models

D3ST The Description-Driven Dialogue Modelling (D3ST) model (Zhao et al., 2022) is a state-of-the-art DST model that performs intent tracking, requested slots prediction, and state tracking in a single pass. See Appendix C for a visual representation of inputs and targets. We process the data and train the model as described in Zhao et al. (2022) and Appendix C, selecting models that maximise the development set JGA.

T5DST We follow Lee et al. (2022) to implement a simplified T5DST (Lee et al., 2021a). It predicts the value of each slot iteratively, requiring a number of decoding passes equal to the number of slots in an API to predict the dialogue state given a dialogue history. Training and inference with this model is very expensive and we train the models with a fixed computational budget of $20,000$ gradient steps[4] for the baseline and $40,000$ steps for all augmented data experiments. See Appendix C for prompt structure and implementation details.

4.2.2 Evaluation

On SGD, the JGA (JGA$_{orig}$) is computed for the $4,201$ test set dialogues. 77% of these have a turn span where the agent calls an API unseen in training. Only 6 out of 21 schemata are seen in training.

Evaluation on SGD-X proceeds as follows. First, the SGD descriptions in the prompt are replaced, in turn, with descriptions taken from the five SGD-X variants. The DST model then predicts the state of a given dialogue 5 times, conditioned on prompts that are increasingly dissimilar to the SGD test set. Hence, the JGA$_{v_{1-5}}$ figures reported are averages over approximately $21,000$ conversations. For all experiments except *oracle* (see Section 4.2.3), *none* of the test time prompts are seen during training:

[4]This is the number of steps required for maximising the development set JGA, for all three runs.

the *seen* superscript in the metric names reported in Section 5 identifies conversations where the *SGD test set prompt is seen* during training. Therefore, it quantifies whether the model can robustly identify slots seen in training by interpreting the meaning of the descriptions rather than relying on linguistic patters in the training schema. Meanwhile *unseen* measures the ability of the model to generalise to new APIs, which may describe new slots and domains, notwithstanding the language used by developers to phrase the descriptions. The JGA coefficient of variation (ie *schema sensitivity*, SS_{JGA}) as the prompt changes measures the sensitivity of a model to the prompt (Lee et al., 2021b).

Metric	Ranking	v1	v2	v3	v4	v5
Jaccard Dist	Pegasus + BART	17.1	63.0	69.5	72.0	76.6
	Backtranslation	18.2	29.9	43.9	-	-
	EDA	3.9	4.1	6.1	16.0	32.9
	SGD-X	55.6	65.6	71.2	78.1	85.7
Entailment	Pegasus + BART	99.0	96.7	94.9	94.6	94.4
	Backtranslation	97.5	96.5	95.9	-	-
	EDA	99.1	98.5	96.6	93.2	86.4
	SGD-X	89.7	88.0	88.4	86.8	87.5
BLEU	Pegasus + BART	13.4	12.5	12.5	13.2	12.7
	Backtranslation	36.4	26.0	18.9	-	-
	EDA	72.0	63.3	47.2	42.3	44.2
	SGD-X	20.4	15.3	10.8	8.3	5.2
self-BLEU	Pegasus + BART	-	12.0	11.4	11.0	10.9
	Backtranslation	-	49.3	41.7	-	-
	EDA	-	87.0	68.8	58.0	53.1
	SGD-X	-	13.5	11.2	9.9	8.6

Table 1: Automatic synthetic schema evaluation. J is multiplied by 100 for readability.

4.2.3 Experimental setup

We show our approach is effective by augmenting the DST training data with synthetic prompts composed from our generated schemata. To study the effect of controlling prompt diversity augmented datasets are two (2x) to six times (6x) the SGD size. For 2x, augmented data contains prompts constructed from the v1 synthetic schema, whereas for 6x we use all five generated schemas[5].

Baselines We create three synthetic schema by **backtranslation**. Our pivot languages are Korean, Japanese and Chinese (Lee et al., 2022). The augmented DST training dataset is four times (4x) larger than SGD. Following Huang et al. (2021), we also consider French and Russian as pivot languages to generate two more synthetic schemas and obtain an augmented dataset six times (6x) larger than SGD. We also compare with **easy data augmentation** (EDA) (Wei and Zou, 2019), a word-level DA approach based on synonym replacement (SR), random insertion, deletion and substitution. We perform SR with probability 0.25 and the other operations with equal probability of 0.05. Just like for backtranslation, we generate 3 or 5 synthetic schemas with this method via the public API. Augmentation with the **SGD-X** human schemata paraphrases is considered an *oracle* because these models see the SGD-X schemata at training time.

[5]Ordering is from most (*v1*) to least (*v5*) similar to SGD.

5 Results and Discussion

5.1 Synthetic schema generation

Our ranking method generates increasingly lexically diverse schemata as shown by the increase in Jaccard distance across schema variants (Table 1). This aspect is much more difficult to achieve with EDA without significantly affecting semantics. Furthermore, self-BLEU (Zhu et al., 2018) scores indicate EDA is the least effective in ensuring candidate diversity compared to other approaches. The BLEU difference between the SGD-X variants v1 and v5 is 15.2 but smaller (0.66) for our approach. Hence, the PEGASUS + BART copies n-grams from the input and includes additional information. This information is not always meaning-preserving: *City where the event is happening* is paraphrased as *The bustling city where the event is taking place* ($v5$) but *End date for the reservation or to find the house* is paraphrased as *End date for hotel reservation to allow time for a replacement both at the struck and in the run up to the event* ($v5$). The self-BLEU of the SGD-X schemas decreases faster compared to the automatically generated paraphrases, suggesting that Jaccard distance increases partly due to hallucination.

Entailment scores show that backtranslation is effective in preserving semantics. For EDA, the semantic similarity drops significantly as more candidates are generated since more dissimilar schemas are generated with more edit operations which are likely to affect meaning. The entailment scores for the SGD-X paraphrases are also lower since they do not always perfectly semantically overlap with the input by construction (Lee et al., 2022) and because of entailment model errors.

5.2 Dialogue state tracking

D3ST Both the robustness and robust generalisation are improved by augmentation with our synthetic schemas, as demonstrated by maximum $\text{JGA}^{seen}_{v_{1-5}}$ (12.35%) and $\text{JGA}^{unseen}_{v_{1-5}}$ (5.85%) increases and 23.6% drop in SS_{JGA} (rows 1&4, Ta-

Model	Index	Generation method - Dataset size	$\text{JGA}_{orig} \uparrow$	$\text{JGA}_{v_{1-5}}$	$\text{JGA}^{seen}_{v_{1-5}}$	$\text{JGA}^{unseen}_{v_{1-5}}$	$\text{SS}_{JGA} \downarrow$
D3ST	1	None - 1x	69.8	56.5	73.6	50.8	70.1
	2	Pegasus + BART - 2x	**72.8**	61.3	80.9	54.8	56.4
	3	Pegasus + BART - 4x	72.6	62.5	81.7	56.1	51.0
	4	Pegasus + BART - 6x	71.2	**63.9**	**85.9**	**56.6**	**46.5**
	5	EDA - 4x (Wei and Zou, 2019)	71.0	59.0	78.5	52.5	63.0
	6	EDA - 6x (Wei and Zou, 2019)	71.4	62.3	83.3	55.3	53.2
	7	Backtranslation - 4x (Lee et al., 2021b)	72.1	62.2	84.0	54.9	53.1
	8	Backtranslation - 6x (Huang et al., 2021)	71.5	61.0	82.5	53.8	54.4
	9	*SGD-X* - 6x (Lee et al., 2021b) *(Oracle)*	73.8	69.7	92.5	62.1	27.9
T5DST	10	None	70.0	50.4	58.5	47.7	87.0
	11	Pegasus B + BART - 4x	71.3	**55.1**	71.2	**49.7**	**70.1**
	12	Pegasus + BART - 6x	68.7	52.5	**71.6**	46.5	77.6
	13	EDA - 6x (Wei and Zou, 2019)	72.2	51.1	55.6	49.6	84.1
	14	Backtranslation - 4x (Lee et al., 2021b)	**72.8**	53.9	67.0	49.6	76.4
	15	*SGD-X* - 6x (Lee et al., 2021b) *(Oracle)*	74.2	67.2	91.8	59.0	36.6

Table 2: SGD and SGD-X dialogue state tracking performance when training with augmented data. Best performance (excluding the oracle setup) is in **bold**. Dataset size is the number of times the augmented dataset is larger than SGD.

ble 2). The most benefit is obtained by training with syntactically diverse prompts (`Pegasus + BART 2x`). Adding more diverse data (rows 3&4) improves DST performance. Part of this improvement may arise because paraphrasing leaves out domain-dependent information: *Average review rating of the doctor* is paraphrased as *The rating is average, so it's not perfect*, so the model can learn to identify ratings more generally[6]. Moreover, inputs are noisy due to hallucination, so the models trained with our augmentation are less likely to overfit to the linguistic patterns of the training schemas. BLEU scores indicate high lexical overlap between EDA-generated and SGD schemas (Table 1). This limits the magnitude of EDA improvement (row 5) and we perform better with less data (rows 3&6, 2&5).

Backtranslation is comparable with our method given the same data quantity (rows 3&7). When we also backtranslate via French and Russian (Huang et al., 2021) the data diversity does not significantly increase (Table 5, Appendix D.1). This negatively impacts the DST performance, while our method improves it (rows 4&8). We can control the schema generation process to match SGD backtranslation performance (Appendix E).

T5DST[7] We outperform EDA (rows 12&13) but not backtranslation (row 14). This may be due to (1) the larger computational budget needed to maximise T5DST performance[8] and (2) T5DST's sensitivity to noisy descriptions owing to its prompt format (Appendix C). We control hallucination by pruning candidates with $J > 0.5$ and entailment smaller than 0.58 and maximise J while minimising S to produce an augmented dataset 4x larger than SGD (`Pegasus B + BART 4x`). Limiting lexical diversity improves entailment compared to `Pegasus + BART 6x` (Table 6, Appendix D.1), and the scheme improves DST robustness compared to the backtranslation baseline (rows 11&14).

The best augmentation schemes fail to improve robustness and generalisation relative to the human baseline (rows 9&15). This is due to the intrinsic challenge of generating diverse yet semantically faithful paraphrases but also due to the fact that humans use common sense and schema information when paraphrasing, so the SGD-X paraphrases are not strictly semantically equivalent. However, the proposed automatic process of paraphrase generation enhances DST, yielding non-trivial improvements in model robustness, while being less costly and more scalable compared to gathering human-written schemata paraphrases.

6 Conclusion and Future Work

We presented a simple tree-based ranking algorithm for optimising lexical diversity and semantic faithfulness during schema generation. The synthetic schemas improve both the DST models' robustness to schemata writing style and their generalisation. Our framework will allow researchers working on paraphrase generation and semantic faithfulness to measure the generalisation of their models in a way that may be difficult to capture by existing benchmarks: it can generate schemata paraphrases and train SOTA dialogue state trackers which were shown to benefit from augmentation with high quality, crowdsourced paraphrases.

[6]It appears in 4 unseen services in the test set.

[7]Lee et al. (2021b) report 72.6% JGA on SGD and 64.0% SGD-X but we could reproduce only 69.98% and 50.42%.

[8]Each training example is seen only once.

Limitations

The optimality of our ranking method depends on the ability of the underlying paraphrase model to generate a search space that contains paraphrases which are lexically and syntactically diverse and preserve the meaning of the input description. This is sometimes challenging with schema inputs which tend to be short (e.g. *name of event*) and contain little information. Our future work will focus on addressing this by contextualising these inputs to enable the paraphrase model to produce a richer space of candidates. Secondly, our method requires that the semantic faithfulness metrics capture semantic similarity well even as the vocabulary of the candidates and their syntax are very diverse. Previous work on abstractive summarisation (Narayan et al., 2022; Maynez et al., 2020; Kryscinski et al., 2019) finds entailment scores to be best correlated with human judgment of faithfulness. However, the correlations are not perfect so the output of the ranking algorithm is still expected to contain noisy candidates. For slot description paraphrases, this is challenging because different inputs are very closely semantically related and the entailment model may not identify paraphrase model errors that map a slot description (e.g. departure time) to one with related semantics (e.g. arrival time). We intend to address this in future work by developing finetuning schemes for semantic faithfulness metrics.

Ethics Statement

Our work is concerned with the use of language generation models to augment training datasets for schema-guided dialogue datasets. The generation phase is unconstrained, so the model may generate candidates that exhibit biases inherited from the C4 (Raffel et al., 2020) and HugeNews (Zhang et al., 2020a) pre-training datasets. In our experiments, we did not observe toxic or harmful outputs, but on one occasion the model did generate the word *apartheid* as part of an incoherent sentence. For this reason, our filtering stack rejects any candidates containing sensitive words. The list of words that parameterize the sensitive words filter is defined by the user.

Acknowledgements

Alexandru Coca was supported supported by EPSRC grant EP/R513180/1. He would like to acknowledge Harrison Lee and Raghav Gupta from Google Research for support and guidance with D3ST and T5DST implementation. Weizhe Lin was supported by a Research Studentship funded by Toyota Motor Europe (RG92562(24020)). We also thank Howard Mei from University of Cambridge for help with editing the final draft. Authors would like to acknowledge the improvement suggestions made by anonymous reviwers and the EACL program committee.

References

Elron Bandel, Ranit Aharonov, Michal Shmueli-Scheuer, Ilya Shnayderman, Noam Slonim, and Liat Ein-Dor. 2022. Quality controlled paraphrase generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 596–609. Association for Computational Linguistics.

Giovanni Campagna, Agata Foryciarz, Mehrad Moradshahi, and Monica S. 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, ACL 2020, Online, July 5-10, 2020*, pages 122–132. 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, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics.

Arash Einolghozati, Sonal Gupta, Mrinal Mohit, and Rushin Shah. 2019. Improving robustness of task oriented dialog systems. *CoRR*, abs/1911.05153.

Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020. Paraphrase augmented task-oriented dialog generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 639–649. Association for Computational Linguistics.

Milan Gritta, Gerasimos Lampouras, and Ignacio Iacobacci. 2021. Conversation graph: Data augmentation, training and evaluation for non-deterministic dialogue management. *Trans. Assoc. Comput. Linguistics*, 9:36–52.

Matthew Henderson, Blaise Thomson, and Steve J. Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the*

SIGDIAL 2014 Conference, The 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue, 18-20 June 2014, Philadelphia, PA, USA, pages 292–299. The Association for Computer Linguistics.

Yutai Hou, Yijia Liu, Wanxiang Che, and Ting Liu. 2018. Sequence-to-sequence data augmentation for dialogue language understanding. In *Proceedings of the 27th International Conference on Computational Linguistics, COLING 2018, Santa Fe, New Mexico, USA, August 20-26, 2018*, pages 1234–1245. Association for Computational Linguistics.

Shuo Huang, Zhuang Li, Lizhen Qu, and Lei Pan. 2021. On robustness of neural semantic parsers. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, EACL 2021, Online, April 19 - 23, 2021*, pages 3333–3342. Association for Computational Linguistics.

Wojciech Kryscinski, Nitish Shirish Keskar, Bryan McCann, Caiming Xiong, and Richard Socher. 2019. Neural text summarization: A critical evaluation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 540–551, Hong Kong, China. 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, EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 7-11 November, 2021*, pages 4937–4949. 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. *CoRR*, abs/2110.06800.

Harrison Lee, Raghav Gupta, Abhinav Rastogi, Yuan Cao, Bin Zhang, and Yonghui Wu. 2022. SGD-X: A benchmark for robust generalization in schema-guided dialogue systems. In *Thirty-Sixth AAAI Conference on Artificial Intelligence, AAAI 2022, Thirty-Fourth Conference on Innovative Applications of Artificial Intelligence, IAAI 2022, The Twelveth Symposium on Educational Advances in Artificial Intelligence, EAAI 2022 Virtual Event, February 22 - March 1, 2022*, pages 10938–10946. AAAI Press.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7871–7880. Association for Computational Linguistics.

Jiexi Liu, Ryuichi Takanobu, Jiaxin Wen, Dazhen Wan, Hongguang Li, Weiran Nie, Cheng Li, Wei Peng, and Minlie Huang. 2021. Robustness testing of language understanding in task-oriented dialog. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, pages 2467–2480. Association for Computational Linguistics.

Samuel Louvan and Bernardo Magnini. 2020. Simple is better! lightweight data augmentation for low resource slot filling and intent classification. In *Proceedings of the 34th Pacific Asia Conference on Language, Information and Computation, PACLIC 2020, Hanoi, Vietnam, October 24-26, 2020*, pages 167–177. Association for Computational Linguistics.

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. *CoRR*, abs/1912.09297.

Joshua Maynez, Shashi Narayan, Bernd Bohnet, and Ryan T. McDonald. 2020. On faithfulness and factuality in abstractive summarization. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 1906–1919. Association for Computational Linguistics.

Biswesh Mohapatra, Gaurav Pandey, Danish Contractor, and Sachindra Joshi. 2021. Simulated chats for building dialog systems: Learning to generate conversations from instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 1190–1203. Association for Computational Linguistics.

Shashi Narayan, Gonçalo Simões, Yao Zhao, Joshua Maynez, Dipanjan Das, Michael Collins, and Mirella Lapata. 2022. A well-composed text is half done! composition sampling for diverse conditional generation. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pages 1319–1339. Association for Computational Linguistics.

Jun Quan and Deyi Xiong. 2019. Effective data augmentation approaches to end-to-end task-oriented dialogue. In *International Conference on Asian Language Processing, IALP 2019, Shanghai, China, November 15-17, 2019*, pages 47–52. IEEE.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *J. Mach. Learn. Res.*, 21:140:1–140:67.

Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. 2020. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 8689–8696. AAAI Press.

Thibault Sellam, Dipanjan Das, and Ankur P. Parikh. 2020. BLEURT: learning robust metrics for text generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7881–7892. Association for Computational Linguistics.

Jason W. Wei and Kai Zou. 2019. EDA: easy data augmentation techniques for boosting performance on text classification tasks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing, EMNLP-IJCNLP 2019, Hong Kong, China, November 3-7, 2019*, pages 6381–6387. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, and Jamie Brew. 2019. Huggingface's transformers: State-of-the-art natural language processing. *CoRR*, abs/1910.03771.

Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, and Qun Liu. 2020. Dialog state tracking with reinforced data augmentation. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9474–9481. AAAI Press.

Jingqing Zhang, Yao Zhao, Mohammad Saleh, and Peter J. Liu. 2020a. PEGASUS: pre-training with extracted gap-sentences for abstractive summarization. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020, 13-18 July 2020, Virtual Event*, volume 119 of *Proceedings of Machine Learning Research*, pages 11328–11339. PMLR.

Yichi Zhang, Zhijian Ou, and Zhou Yu. 2020b. Task-oriented dialog systems that consider multiple appropriate responses under the same context. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9604–9611. AAAI Press.

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. *CoRR*, abs/2201.08904.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, SIGIR 2018, Ann Arbor, MI, USA, July 08-12, 2018*, pages 1097–1100. ACM.

A The SGD and SGD-X datasets

SGD As mentioned in Section 1, the conversations in the SGD dataset are grounded in schemata, which describe a set of service APIs. The most important schema *elements* are[9]:

- a *service name* (e.g. *Messaging_1*) followed by a *service description* (e.g. *Connect and share locations with your contacts*)
- one or more API functions to be invoked as users solve tasks, referred to as *(user) intents*; each intent has a *name* (e.g. *ShareLocation*) and an *intent description* (e.g. *Send your location to a contact*)
- optional and required arguments for each API function, or *slots*; each slot has a *name* (e.g. *location*) and a *slot description* (e.g. *Location to share with the contact*)

SGD-X Lee et al. (2022) observe that 71% of intent names and 65% of slot names from unseen APIs exactly match the train set. Furthermore, descriptions are stylistically uniform across the train and test sets. For example, all boolean slots begin with the phrase *Boolean flag ...* or *Whether...*. Therefore, they create the SGD-X dataset as follows:

- crowdsource schema element paraphrasing to more than 400 authors via Amazon Mechanical Turk. Each crowdworker either paraphrases all names or all descriptions for a given schema

[9]Examples below are taken from the SGD test set.

- manually vet responses for quality and correctness.

The slot names collected are sorted in increasing order of their Levenshtein distance to the SGD slot names whereas the descriptions are sorted according to the Jaccard distance between their lemmatized forms (excluding stop words). An example of SGD-X description paraphrases in shown in Table 3.

Variant	Description
SGD	Category to which the attraction belongs
v1	The category that describes what kind of attraction it is
v2	Category of place of interest
v3	Type of tourist attraction
v4	Choose the kind of tourist landmark
v5	The kind of tourist hotspot

Table 3: Example of descriptions paraphrases from the SGD-X test schemas. The more similar `v1` description contains overlapping vocabulary with the SGD test set description, whereas `v4` and `v5` variants are dissimilar both stylistically and lexically

While the examples above are paraphrases of the SGD input, in general, the semantic content of the schema element paraphrases is not perfectly overlapping with the input as the crowdworkers use information from the wider service context when creating new elements.

B Ranking Framework

B.1 Candidate generation

Algorithm 2 summarises the candidate generation procedure, which takes any paraphrase model, a list of model-specific generation parameters and, optionally, a list of filters as an input (line 1). These parameters are temperature and number of beams for Pegasus, or a grid of lexical, semantic and syntactic distances for the Quality Controlled Paraphrase Generation (QCPG) (Bandel et al., 2022) model presented in Appendix E. The model generates one or more paraphrases, which are filtered before returning (lines 3-8). We describe the filtering process next.

Heuristic filtering Our main motivation for implementing heuristic filters is to filter the majority of poor quality candidates, without making use of the large GPU cards required to run the entailment model. We also address the fact that the model is free to generate a very large number of candidates and therefore is expected to hallucinate significantly. These filters are general purpose and are implemented in few lines of code using the `spaCy` and `nltk` libraries. Table 4 lists active filters along with typical examples filtered.

Entailment filtering We implement our entailment filter using BART (Lewis et al., 2020)[10]. This model is pre-trained on the MNLI dataset (Williams et al., 2018). To measure entailment this model consumes a premise and hypothesis in the format `premise <SEP> hypothesis`. In our implementation we replace `premise` with the description to be paraphrased. By default, the `hypothesis` is a template of the form `This example is {}.`, where `{}` is a placeholder for the user hypothesis, in our case the paraphrased description. We find that considering alternative templates improves the reliability of the model, so we consider `{}`, `This example has the same meaning as {}.`, `This text is about {}.`, and `This example implies that {}.`, averaging the entailment scores across templates to calculate the entailment score. The same procedure is followed when computing the entailment of candidates during ranking.

Algorithm 2: Candidate generation

```
1: def generate_candidates(model: Any,
     inp: str, params: dict, filters:
     Optional[list[Callable]]):
       Data: model text generation model, inp
             input sentence, params model
             specific parameters, filters a
             list of boolean functions
       Result: cands list of inp paraphrases
2:     cands ← [] ;
3:     for p in params:
4:         c ← model.forward(inp, **p) ;
5:         c ← [p for p in c if not any(f(p,
            inp) for f in filters)]
6:         cands.extend(c)
7:     return cands
```

B.2 Ranking

Ranking Algorithm 3 summarises the tree-ranking procedure. This procedure takes as an input the tree constructed as described in Algorithm 1, along with a list of decision functions `f_dec`. Our algorithm starts by selecting a paraphrase via depth first traversal of the subtree rooted at $J = 0$ (line 2). The remainder of the candidates are selected by

[10]Avaialble at https://huggingface.co/facebook/bart-large-mnli.

Filter name	Filtered example
`contains advice`	An appointment **is necessary for your hair**.
`describes action`	They commemorate the number of flights to the airport.
`has named entities`	Enter the doctor's **Leningrad** address.
`has low frequency words`	The address is **ofadvisory**.
`discard multiple sentences`	The address is the dentist's box. Guidelines for hiring a dentist.
`has repeated ngrams`	**The dentist** is Address of **the dentist**.
`has repeated similar bigrams`	The **type of event** is stated in the title **of the event**.
`has consecutive repeated words`	Average review rating for a hotel hotel.
`is past tense sentence`	It was the dentist's address.
`is passive voice sentence`	The address was given by abrasives from the dentist.
`is question`	Is there a balance of the account?
`has alphanumeric words`	**400** baths in an apartment.

Table 4: Filters implemented along with sample examples they discard

Algorithm 3: Tree ranking

```
1:  def tree_rank(root: Node, n: int, f_dec: list[Callable]):
        Data: root, n number of candidates,
              f_dec decision functions
        Result: list of n ranked candidates
2:      ranked ← syntax_select (root) ;
3:      n ← n - len(ranked) ;
4:      while len(ranked) ≠ n:
5:          for next in level_order (root):
6:              for f in f_dec:
7:                  cand ← select (next.sents) ;
8:                  ranked.add(cand) ;
9:                  prune (next, cand) ;
10:     return ranked
```

traversing the first level in a breadth-first manner (line 5) and depth-first traversal of each subtree returned during the level-order traversal (lines 6-8). Here the semantics of `f(next.children)` is that the decision function `f` takes all the children of `next` as input and returns a single node which is next in the traversal. A candidate is selected from the leaf[11](line 8) and subsequently removed from the candidates list (line 10). This is to avoid selecting the same candidate multiple times in situations where the paraphrase model generates few distinct candidates.

C State Tracking Baselines

D3ST We process the data as described by Zhao et al. (2022) with the following differences:

- The indices are separated by the = symbol in both the inputs and the targets, to avoid a parsing ambiguity which occurs for time slots if : is used as a separator for targets

- For categorical slots which take the `dontcare` special value, our output contains `slot_index: dontcare` substring and we do not include the `dontcare` value in the prefix together with the other options

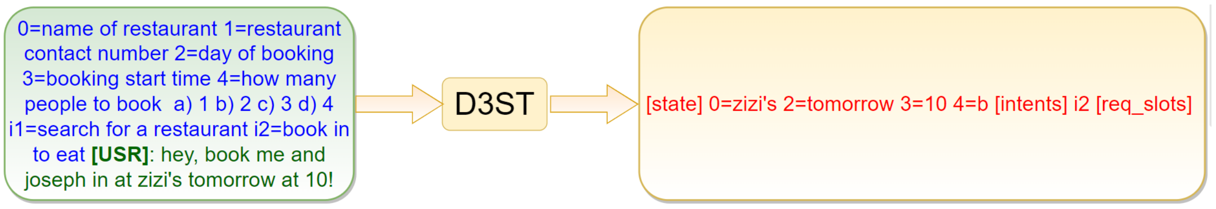
- We lowercase the inputs and the targets[12].

We obtain $175,780$ examples from the original SGD dataset, which are truncated to the last $1,024$ tokens on the input side. See Figure 2a for a visual representation of the model inputs and outputs. We optimise the model using the Adafactor optimizer and effective batch size 32, starting from the initial weights `google/t5-v1_1-base` published by `huggingface` (Wolf et al., 2019). We interpolate the learning rate linearly between 0 and 10^{-4} over the first 1000 steps and keep it constant thereafter. We select the model by evaluating the development set JGA every 5000 gradient updates, stopping the training if said metric fails to improve after 3 consecutive evaluations. All numbers in Table 2 are averages of 3 runs, except the SGD-X experiment for T5DST which is a single run.

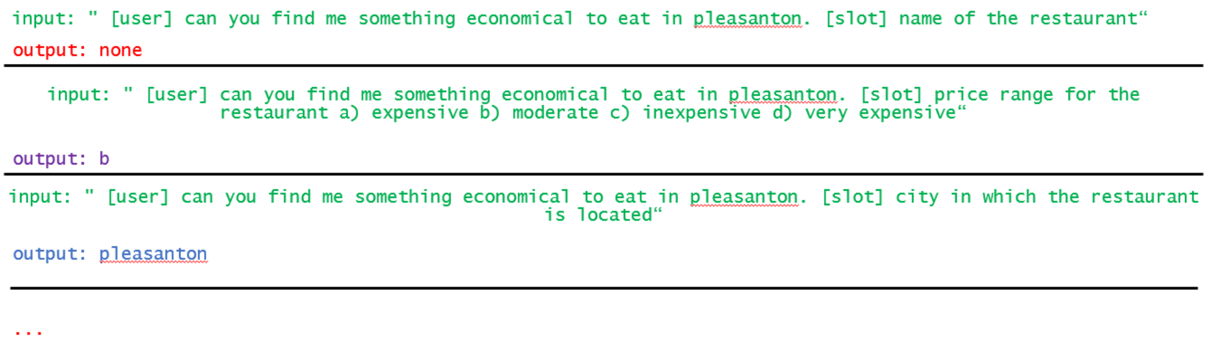
T5DST Given a dialogue in the SGD training set we consider all partial dialogue histories $\{u_1, s_1, ... s_{t-1}, u_t\}$ with $t \in \overline{0, T}$ where T is maximum index of the user turn in a dialogue. The turns in each dialogue history are lowercased and separated `[usr]` and `[sys]` tokens, *not treated* as special tokens. For each dialogue history we create a training example for

[11]There can be multiple, possibly repeated candidates in a leaf because the generative model may generate the same output given different parameter settings. We select the most common one if there are repeated candidates and randomly otherwise.

[12]This appears in illustrations but is not explicitly stated by (Zhao et al., 2022).


0=name of restaurant 1=restaurant contact number 2=day of booking 3=booking start time 4=how many people to book a) 1 b) 2 c) 3 d) 4 i1=search for a restaurant i2=book in to eat **[USR]**: hey, book me and joseph in at zizi's tomorrow at 10!
D3ST
[state] 0=zizi's 2=tomorrow 3=10 4=b [intents] i2 [req_slots]

(a) Visual representation of D3ST inputs and targets. Blue font blue, preceding the `[USR] special token represents the prompt, consisting of slot descriptions extracted from the schema. Each slot is assigned an index, which is used to recover the slot value pairs during post-processing. Note that slot 4 is categorical, and so the string a) 1 b) 2 c) 3 d) 4` is appended to the description to indicate the model that it should output one of the choices. The dialogue history follows the `[USR]` token. The model outputs only active slots, in this case omitting slot 1 because is has not been mentioned. The entire dialogue state (as opposed to turn state) is generated at every turn.



```
input: " [user] can you find me something economical to eat in pleasanton. [slot] name of the restaurant“
output: none

input: " [user] can you find me something economical to eat in pleasanton. [slot] price range for the
          restaurant a) expensive b) moderate c) inexpensive d) very expensive“

output: b

input: " [user] can you find me something economical to eat in pleasanton. [slot] city in which the restaurant
          is located“

output: pleasanton

...
```

(b) Visual representation of T5DST inputs and targets. The string `[slot]` separates the dialogue history from the slot description. In the first example the model outputs the special value none to indicate that the slot was not mentioned by the user. For categorical slots (second row from the top), the slot description is concatenated with options containing all possible slot values and the model predicts the correct option. For non-categorical slots (third row), the exact value is predicted. The ellipsis indicates that none is predicted for all other slots in the `Restaurant_1` schema that are not mentioned in the dialogue history

Figure 2: Prompt formats for a) D3ST b) T5DST

each slot in the ground truth schema, which contains the concatenated turns suffixed with the string `[slot] [slot_description]` where the placeholder `[slot_description]` is replaced by the lowercase descriptions extracted from the SGD schemata. This yields $1,601,356$ examples for the SGD training dataset. See Figure 2b for a representation of the model inputs and outputs.

We optimise the model using the Adafactor optimizer and effective batch size 256, starting from the initial weights `google/t5-v1_1-base` published by `huggingface`. We interpolate the learning rate linearly between 0 and 10^{-4} over the first 1000 steps and keep it constant thereafter. We perform $20,000$ optimisation steps[13], limiting the number of training steps to $40,000$ steps[14] for all augmented data experiments. All numbers in Table 2 are averages of 3 runs, except the Oracle experiment on T5DST which is a single run.

[13]For a single run, this is approximately 6 hours of computation on 8 `nvidia A100-80GB` cards. Moreover, decoding a single run on SGD and SGD-X takes 6 hours.

[14]This is sufficient so that the model sees every example once when working with an augmented training set six times the size of SGD.

D Additional Results

D.1 Increasing backtranslation dataset size

We include Table 5 to substantiate our intuition that the training with the Backtranslation 6x scheme does not yield further improvement compared to the Backtranslation 4x scheme as the additional data does not significantly increase the prompt diversity. Most clearly, this is indicated by the fact that the $v5$ variant has similar BLEU to variant $v3$ in Backtranslation 4x, indicating that a large proportion of additional data has some overlaps more with the SGD distribution than the data backtranslated to Chinese, Korean and Japanese. This is also indicated by how self-BLEU decays as more data is added, comparatively, between `Backtranslation 4x` and `Backtranslation 6x`.

D.2 Controlling schema generation diversity

Table 6 shows that the alternative schema scheme generates schemas with lower average Jaccard distance and higher entailment with respect to the SGD schemata. We find this effectively controls the noise in the data, leading to improved performance

Metric	Ranking	v1	v2	v3	v4	v5
Jaccard Dist	Backtr. 4x	18.2	29.9	43.9	-	-
	Backtr. 6x	12.9	22.7	27.8	35.6	46.7
Entailment	Backtr. 4x	97.5	96.5	95.9	-	-
	Backtr. 6x	98.0	97.5	95.2	94.8	95.5
BLEU	Backtr. 4x	36.4	26.01	18.9	-	-
	Backtr. 6x	51.3	37.2	29.5	23.4	18.2
self-BLEU	Backtr. 4x	-	49.3	41.7	-	-
	Backtr. 6x	-	55.3	49.7	44.6	39.6

Table 5: Effect of using French and Russian as additional pivot languages on automatic metrics

Metric	Ranking	v1	v2	v3	v4	v5
Jaccard Dist	Pegasus + BART	13.0	61.2	68.8	71.2	76.2
	Pegasus B + BART	10.2	38.3	46.9	55.1	54.5
	SGD-X	55.6	65.6	71.2	78.1	85.7
Entailment	Pegasus + BART	99.1	96.3	94.6	94.2	94.2
	Pegasus B + BART	98.8	98.2	96..4	96.2	96.7
	SGD-X	89.7	88.0	88.4	86.8	87.5

Table 6: Comparison of diversity and semantic faithfulness metrics for slot description paraphrases

Index	Augmentation	JGA$_{orig}$	JGA$_{v1-5}$	JGA$^{seen}_{v1-5}$	JGA$^{unseen}_{v1-5}$	SS$_{JGA}$
1	Pegasus+BART 6x	71.2	<u>63.9</u>	<u>85.9</u>	**56.6**	**46.5**
2	Pegasus+BLEURT 6x	<u>72.4</u>	**64.0**	**86.6**	<u>56.4</u>	<u>46.6</u>
3	QCPG+BLEURT 6x	**72.7**	63.2	85.2	55.9	47.3
4	Backtranslation 4x (Lee et al., 2021b)	72.1	62.2	84.0	54.9	53.1

Table 7: Ranking with a more accurate semantic faithfulness metric (row 2) or generating candidates with a controllable paraphrase model (row 4) can be used to boost SGD performance over our Pegasus+BART approach (row 1). Bold font marks column maximum, underlined second largest number.

for T5DST and similar performance to PEGASUS + BART for D3ST.

E Schema Generation with BLEURT and QCPG

BLEURT (Sellam et al., 2020) is a BERT-based natural metric commonly used in translation, so it is expected to be highly sensitive to semantic differences. In Table 7 we show that simply re-ranking the Pegasus output space with BLEURT improves SGD performance comparably with backtranslation (rows 2&4) and the robustness and generalisation improvements are maintained.

Bandel et al. (2022) exploit high quality examples in paraphrase corpora by conditioning the model with a string *quality parameters string* outlining target semantic, syntactic and lexical distances of the generated paraphrase during finetuning. At inference one must specify these parameters to obtain diverse yet high quality paraphrases. We could not apply the quality parameter selection method proposed by QCPG authors at inference time as the code had not been fully released at the time of writing. Instead, we generated a large number of paraphrases with different quality targets and greedy decoding, and re-ranked the candidates using our framework. This demonstrates the versatility of our framework. In Table 7 we show that this model can equally achieve improved performance on SGD. The improvement on SGD-X is slightly less than achieved by PEGASUS+BART 6x, as expected since greedy decoding and better semantic faithfulness optimisation generate schemata closer to the SGD distribution so less out-of-distribution improvement is achieved.

This experiments in this section and Appendix D.2 demonstrate the versatility of our framework and its usefulness as a tool for generating synthetic schema prompts.