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 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. 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-

Variants are ordered according to their lexical similarity to the SGD schemata. 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, \( m_\text{val} \), and the candidate paraphrases which are split at that node, \( \text{sents} \). 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, 
10:                     sent=[c]);
11:                 curr.children.add(next)
12:             else:
13:                 next.sents.add(c);
14:                 curr ← next
15:     return root
```

Ranking Our ranker input is the tree and a list of decision functions, with elements corresponding to each level in the tree, \( f_\text{dec} \). 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 \( \text{max} \) decision function to the children of \( J = 66 \), and that applying \( \text{min} \) 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/3vgY7EY.
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 $J_{\text{dec}} = \{\text{None}, \text{max}, \text{min}\}$. 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 breadth-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 $v^1$ being the most similar to the SGD schema and $v^5$ 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 ($JGA_{\text{orig}}$) 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 $JGA_{\text{orig}}$ 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 \textit{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 \textit{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 \textit{schema sensitivity}, \textit{SS}_{JGA}) as the prompt changes measures the sensitivity of a model to the prompt (Lee et al., 2021b).

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 \textit{PEGASUS + BART} copies \textit{n}-grams from the input and includes additional information. This information is not always meaning-preserving: \textit{City where the event is happening} is paraphrased as \textit{The bustling city where the event is taking place} (v5) but \textit{End date for the reservation} or to \textit{find the house} is paraphrased as \textit{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 schemata 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 schemata 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

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

\begin{table}
\centering
\begin{tabular}{|c|c|c|c|c|c|}
\hline
\textbf{Metric} & \multicolumn{5}{c|}{\textbf{Ranking}} \\
\hline
 & v1 & v2 & v3 & v4 & v5 \\
\hline
\textit{Jaccard Dist} & \text{Pegasus + BART} & 17.1 & 63.0 & 69.5 & 72.0 & 76.6 \\
 & \text{Backtranslation} & 18.2 & 29.9 & 43.9 & - & - \\
 & \text{EDA} & 3.9 & 4.1 & 6.1 & 16.0 & 32.9 \\
 & \text{SGD-X} & 55.6 & 65.6 & 71.2 & 78.1 & 85.7 \\
\hline
\textit{Entailment} & \text{Pegasus + BART} & 99.0 & 96.7 & 94.9 & 94.6 & 94.4 \\
 & \text{Backtranslation} & 97.5 & 96.5 & 95.9 & - & - \\
 & \text{EDA} & 99.1 & 98.5 & 96.6 & 93.2 & 86.4 \\
 & \text{SGD-X} & 89.7 & 88.0 & 88.4 & 86.8 & 87.5 \\
\hline
\textit{BLEU} & \text{Pegasus + BART} & 13.4 & 12.5 & 12.5 & 13.2 & 12.7 \\
 & \text{Backtranslation} & 36.4 & 26.0 & 18.9 & - & - \\
 & \text{EDA} & 72.0 & 63.3 & 47.2 & 42.3 & 44.2 \\
 & \text{SGD-X} & 20.4 & 15.3 & 10.8 & 8.3 & 5.2 \\
\hline
\textit{self-BLEU} & \text{Pegasus + BART} & - & 12.0 & 11.4 & 11.0 & 10.9 \\
 & \text{Backtranslation} & - & 49.3 & 41.7 & - & - \\
 & \text{EDA} & - & 87.0 & 68.8 & 58.0 & 53.1 \\
 & \text{SGD-X} & - & 13.5 & 11.2 & 9.9 & 8.6 \\
\hline
\end{tabular}
\caption{Automatic synthetic schema evaluation. \textit{J} is multiplied by 100 for readability.}
\end{table}

\textbf{Baselines} We create three synthetic schema by \textbf{backtranslation}. Our pivot languages are Korean, Japanese, and Chinese (Lee et al., 2022). The augmented \textit{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 \textbf{easy data augmentation} (EDA) (Wei and Zou, 2019), a word-level DA approach based on synonym replacement

\footnote{Ordering is from most (v1) to least (v5) similar to SGD.}
We outperform EDA (rows 12&13) but backtranslation (row 14). This may be 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.
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 by EP-SRC grant EP/R513180/1. He would like to acknowledge Harrison Lee and Raghav Gupta from Google Research for support and guidance with D3ST and TSDST 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 reviewers and the EACL program committee.

References


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.


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.


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


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:

- 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:

- crowdsourcing 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

9Examples 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.

<table>
<thead>
<tr>
<th>Variant</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>SGD</td>
<td>Category to which the attraction belongs</td>
</tr>
<tr>
<td>v1</td>
<td>The category that describes what kind of attraction it is</td>
</tr>
<tr>
<td>v2</td>
<td>Category of place of interest</td>
</tr>
<tr>
<td>v3</td>
<td>Type of tourist attraction</td>
</tr>
<tr>
<td>v4</td>
<td>Choose the kind of tourist landmark</td>
</tr>
<tr>
<td>v5</td>
<td>The kind of tourist hotspot</td>
</tr>
</tbody>
</table>

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 (QCPCG) (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)\(^\text{10}\). 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

```python
1: def generate_candidates(model: Any, inp: str, params: dict, filters: Optional[set[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 \( f_{\text{dec}} \). 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

\(^{10}\)Available at https://huggingface.co/facebook/bart-large-mnli.
Table 4: Filters implemented along with sample examples they discard

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

Algorithm 3: Tree ranking

```python
1: def tree_rank(root: Node, n: int, f_dec: list[Callable]):
2:     Data: root, n number of candidates, f_dec decision functions
3:     Result: list of n ranked candidates
4:     ranked ← syntax_select (root);
5:     n ← n - len(ranked);
6:     while len(ranked) ≠ n:
7:         for next in level_order (root):
8:             for f in f_dec:
9:                 cand ← select (next.sents);
10:                ranked.add(cand);
11:               prune (next.cand);
12:          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 \( f(\text{next.children}) \) 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 \( \times 10^{-4} \) 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 \( t \in 0, T \) 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 each slot in the ground truth schema, which contains the concatenated turns suffixed with the string

\(^{11}\)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).
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.

\[\text{Reference code: } \text{Figure 2: Prompt formats for a) D3ST b) T5DST}\]
Table 5: Effect of using French and Russian as additional pivot languages on automatic metrics

<table>
<thead>
<tr>
<th>Metric</th>
<th>Ranking</th>
<th>v1</th>
<th>v2</th>
<th>v3</th>
<th>v4</th>
<th>v5</th>
</tr>
</thead>
<tbody>
<tr>
<td>Jaccard Dist</td>
<td>Backtr. 4x</td>
<td>18.2</td>
<td>29.9</td>
<td>43.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Backtr. 6x</td>
<td>12.9</td>
<td>22.7</td>
<td>27.8</td>
<td>35.6</td>
<td>46.7</td>
</tr>
<tr>
<td>Entailment</td>
<td>Backtr. 4x</td>
<td>97.5</td>
<td>96.3</td>
<td>95.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Backtr. 6x</td>
<td>98.0</td>
<td>97.5</td>
<td>95.2</td>
<td>94.8</td>
<td>95.5</td>
</tr>
<tr>
<td>BLEU</td>
<td>Backtr. 4x</td>
<td>36.4</td>
<td>26.0</td>
<td>18.9</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Backtr. 6x</td>
<td>51.3</td>
<td>37.2</td>
<td>29.5</td>
<td>23.4</td>
<td>18.2</td>
</tr>
<tr>
<td>self-BLEU</td>
<td>Backtr. 4x</td>
<td>-</td>
<td>49.3</td>
<td>41.7</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td></td>
<td>Backtr. 6x</td>
<td>-</td>
<td>55.3</td>
<td>49.7</td>
<td>44.6</td>
<td>39.6</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Index</th>
<th>Augmentation</th>
<th>JGA_{orig}</th>
<th>JGA_{seen}^{1-5}</th>
<th>JGA_{unseen}^{1-5}</th>
<th>SS_{JGA}</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Pegasus+BART 6x</td>
<td>71.2</td>
<td>63.9</td>
<td>85.9</td>
<td>56.6</td>
</tr>
<tr>
<td>2</td>
<td>Pegasus+BLEURT 6x</td>
<td>72.4</td>
<td><strong>64.0</strong></td>
<td><strong>86.6</strong></td>
<td>56.4</td>
</tr>
<tr>
<td>3</td>
<td>QCPG+BLEURT 6x</td>
<td>72.7</td>
<td>63.2</td>
<td>85.2</td>
<td>55.9</td>
</tr>
<tr>
<td>4</td>
<td>Backtranslation 4x (Lee et al., 2021b)</td>
<td>72.1</td>
<td>62.2</td>
<td>84.0</td>
<td>54.9</td>
</tr>
</tbody>
</table>

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.