Diable: Efficient Dialogue State Tracking as Operations on Tables

Pietro Lesci[1]* **Yoshinari Fujinuma**[2] **Momchil Hardalov**[2]
Chao Shang[2] **Yassine Benajiba**[2] **Lluís Màrquez**[2]
[1]University of Cambridge [2]AWS AI Labs
pl487@cam.ac.uk;
{fujinuy,momchilh,chshang,benajiy,lluismv}@amazon.com

Abstract

Sequence-to-sequence state-of-the-art systems for dialogue state tracking (DST) use the full dialogue history as input, represent the current state as a list with all the slots, and generate the entire state from scratch at each dialogue turn. This approach is inefficient, especially when the number of slots is large and the conversation is long. We propose *Diable*, a new task formalisation that simplifies the design and implementation of efficient DST systems and allows one to easily plug and play large language models. We represent the dialogue state as a table and formalise DST as a table manipulation task. At each turn, the system updates the previous state by generating table operations based on the dialogue context. Extensive experimentation on the MultiWoz datasets demonstrates that *Diable (i)* outperforms strong efficient DST baselines, *(ii)* is 2.4x more time efficient than current state-of-the-art methods while retaining competitive Joint Goal Accuracy, and *(iii)* is robust to noisy data annotations due to the table operations approach.

efficient-dialogue-state-tracking-by-sequential-information-processing

1 Introduction

Dialogue state tracking (DST; Jacqmin et al., 2022) is the task of tracking user requests from the dialogue history in the form of slot-value pairs (Henderson et al., 2014; Mrkšić et al., 2015; Rastogi et al., 2020a). The slots are defined in a domain-specific schema and represent the fields that need to be extracted from the dialogue to execute queries in the backend and generate responses. Recent generative approaches to DST based on language models (Wu et al., 2019; Kim et al., 2020) often use the entire dialogue history as input and represent the state, at each turn, as the concatenation of all the slots in the schema, where inactive slots are reported with a placeholder value (see Figure 2). This representation is known as *cumulative state* (Hosseini-Asl et al., 2020; Feng et al., 2021; Zhao et al., 2022) and implies the generation of all the states from scratch at each dialogue turn. This approach is computationally inefficient, especially for long conversations and large schemas.

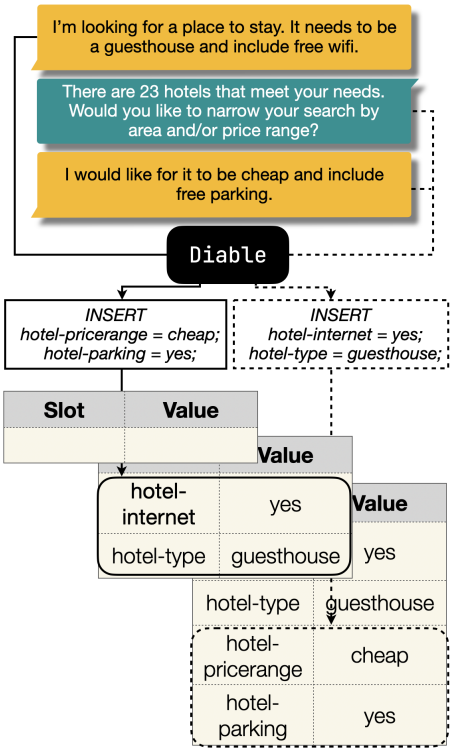

Figure 1: *Diable* approach to DST. The figure presents the first two turns of a dialogue (user's utterances are orange, system's are green). When the conversation starts, the state table is empty. At each dialogue turn, the system outputs a table update operation (either `INSERT` or `DELETE`), and the state is modified accordingly.

We propose Efficient Dialogue State tracking as Operations on Tables (*Diable*, shown in Figure 1), a novel task formulation and a new DST approach that better uses the generative capabilities of language models. Our approach simplifies the

*Work conducted during an internship at AWS AI Labs.

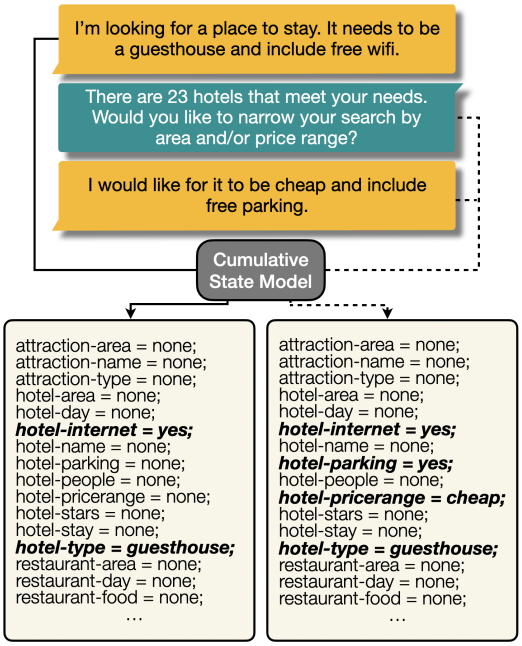

Figure 2: Cumulative state approach to DST. At each dialogue turn, the system outputs all the slots. Inactive slots are filled with a placeholder value (none).

design and implementation of DST systems and works with any sequence-to-sequence model. Our intuition is that a DST system translates conversations into filters for database searches. Inspired by formal languages for databases and the recent success in applying sequence-to-sequence models to text-to-SQL tasks (Yin et al., 2020; Scholak et al., 2021), we represent the dialogue state as an implicit table and frame DST as a table manipulation task. At each turn, the system updates the previous state by generating update operations expressed in a simplified formal language based on the current dialogue context (see Figure 1). *Diable* is the first end-to-end DST system that outputs state operations and values jointly while processing all slots simultaneously.

Based on extensive experimentation using the MultiWoz benchmark (Budzianowski et al., 2018), we show that *Diable* can successfully and efficiently translate conversations into filters for database searches. Our approach minimises the number of input and output tokens required resulting in a significant efficiency gain (2.4x reduction in inference time compared to state-of-the-art cumulative state systems).

Our main contributions are as follows:

- We introduce a novel DST task formulation and a new system, *Diable*, specifically designed to enhance efficiency and leverage the capabilities of state-of-the-art sequence-to-sequence models.
- We show that our DST task formulation does not require ad-hoc data preprocessing, the full history, or extra supervision and works with any sequence-to-sequence model without requiring any architectural modification.
- We demonstrate that *Diable* achieves better Joint Goal Accuracy on MultiWoz than other efficient baselines while being competitive with state-of-the-art cumulative state systems.
- We show that *Diable* is robust to noise in the training data, resulting in more stable results across three versions of MultiWoz.

2 A Taxonomy of DST Approaches

The goal of DST systems is to handle long, diverse conversations in multiple domains with large schemas and unrestricted vocabulary, potentially without extra supervision (Eric et al., 2020; Rastogi et al., 2020b). Achieving this goal has prompted the development of different DST approaches.

Ontology-based approaches *treat DST as either a classification or a token classification task. They assume that all possible slot-value pairs are restricted to a fixed set, or ontology, either predefined or extracted from the training data. Classification-based approaches output a probability distribution over values given the dialogue context and a slot (Henderson et al., 2014) while token classification approaches output a probability distribution over slots for each token (Liao et al., 2021).*
The ontology-based formulation simplifies the DST task considerably, thus the performance of these systems is usually relatively high for specific datasets (Zhong et al., 2018; Ye et al., 2021, 2022a). Complex dialogues with large schemas pose a significant challenge for traditional ontology-based approaches as they do not easily generalise to new domains nor scale to large ontologies (Mrkšić et al., 2017; Rastogi et al., 2017; Zhong et al., 2018; Ye et al., 2021). For this reason, ontology-based approaches are out of scope for our paper.

Open-vocabulary approaches *address these limitations by formulating DST as either a reading comprehension task wherein for each slot a span is extracted from the dialogue context (Gao et al., 2019; Chao and Lane, 2019), or as a generation task wherein a value for each slot is generated*

based on the dialogue history (Wu et al., 2019).

By leveraging sequence-to-sequence models (Brown et al., 2020; Lewis et al., 2020; Raffel et al., 2020), generative approaches have recently achieved states-of-the-art results (Xu and Hu, 2018; Lee et al., 2019; Chao and Lane, 2019; Gao et al., 2019; Wu et al., 2019; Kumar et al., 2020; Heck et al., 2020; Hosseini-Asl et al., 2020; Lee et al., 2021; Zhao et al., 2021, 2022). However, these methods predict the dialogue state from scratch at each turn and generate a value for each slot, even when a slot is not active (Figure 2). We argue (§6) that these are the main sources of inefficiencies of current DST systems. We compare *Diable* with these methods in the "Cumulative State Models" section of Table 1.

Efficient approaches *seek efficiency by minimising the number of values to generate, thus decomposing DST into two successive sub-tasks: state operation prediction and value generation. In this way, only the slots that need to be changed are considered for value generation (Kim et al., 2020).*

These approaches are the most related to *Diable* in that they target efficiency. We compare them against *Diable* in section "Efficient Models" of Table 1. Often, these methods (Ren et al., 2019; Zhu et al., 2020) use the cumulative state representation which is the primary source of inefficiencies (we discuss this issue in the context of §5, Table 2) and need to output operations for all slots.

For example, Kim et al. (2020) and Zeng and Nie (2020) predict an operation for each slot in the input by adding a classification head on top of the tokens representing the individual slots and predict four kinds of state operations: "carryover", "delete", "dontcare", and "update". For those slots categorised as "update", the contextual representation is further processed to decode the slot value. However, this approach limits the ability of such systems to deal with large schemas because the full schema needs to fit in the input context. Differently from these approaches, we remove the two-component structure by adopting a sequence-to-sequence approach that allows us to jointly generate operations and values for all slots simultaneously and works with any sequence-to-sequence model. Importantly, we only need to predict operations for the active slots (i.e., the slots actually mentioned in the conversation).

Lin et al. (2020b) seek efficiency differently by introducing the notion of "Levenshtein belief span". Based on the concept of *belief span* (Lei et al., 2018) that reformats the dialogue state into a text span allowing models to generate slot values dynamically. They propose to only focus on the *differences* between states at subsequent turns. We take this approach a step further by explicitly outputting operations for all slots changing from one turn to another simultaneously while retaining our minimal state representation.

3 *Diable*: Dialogue State Tracking as Operations on Tables

We introduce a novel efficient formulation of the DST task and a system, *Diable*, specifically designed to enhance efficiency and optimise the capabilities of state-of-the-art sequence-to-sequence models. In this section, we describe our approach, formalise the DST problem, and introduce the concepts of *state as a table* and *state operations*.

3.1 Problem Definition

The goal of DST is defined as learning a mapping from a dialogue context to a dialogue state. Specifically, let $D_{1:T} = (D_1, \ldots, D_T)$ denote a dialogue of T turns, where $D_t = (U_t, R_t)$ represent an utterance composed of the user query and system response at turn t, respectively. At turn t, the dialogue context, C_t, is defined as the set of all the information available up to that turn. It always includes the current dialogue utterance, but can additionally contain the previous state, utterances from the dialogue history, and extra supervision (e.g., slot descriptions and the schema). We consider a dialogue context composed by only the previous dialogue turn(s) and the previous state, that is $C_t = (D_t, \mathcal{B}_{t-1})$. We do not use any schema information and let the model learn it during training.[1] The dialogue state at turn t is defined as a set $\mathcal{B}_t = \{(s, v_t) | s \in \mathcal{S}_t\}$, where $\mathcal{S}_t \subseteq \mathcal{S}$ denotes the subset of active slots at that turn out of all the predefined slots in the schema and v_t is the value corresponding to slot s.

3.2 The *Diable* Approach

In our approach, instead of directly outputting the dialogue state $\mathcal{B}$, we learn a mapping from the dialogue context, C, to a set of operations $\mathcal{O}$. At the beginning of each conversation the state table,

[1] Our preliminary study showed that passing the schema to the model has little effect on performance but hurts the model's efficiency as it needs to encode more tokens.

$\mathcal{B}_0$, is initialised empty. At turn t, based on the dialogue context, C_t, the DST system generates the set of operations, $\mathcal{O}_t = \{O_1, \ldots, O_{N_t}\}$, where N_t is the number of slots that change between turn $t-1$ and t. Finally, the generated operations are applied to the previous state to get the new state. The tracker is expected to carry over the previously extracted slots into the current state, i.e., the state at each turn includes all the slots active since the beginning of the conversation up to that point.

We operationalise this process by framing it as a sequence-to-sequence task in which a model, f_θ, receives a textual representation of the dialogue context, $\tau_c(C_t)$, and outputs a textual representation of the operations needed, $\tau_s(\mathcal{O}_t)$, where τ_c and τ_s are the templating functions that convert the dialogue context and state operations to a string, respectively. We provide more details about these functions in Appendix B. The structure of the system can be described as follows

$$\tau_s(\mathcal{O}_t) = f_\theta(\tau_c(C_t)) \quad (1)$$

$$\mathcal{B}_t = \text{Interpreter}(\tau_s(\mathcal{O}_t), \mathcal{B}_{t-1}) \quad (2)$$

where in Eq. (2) we use an operation interpreter to parse the string representation of the operations and apply them to the previous state. Based on the definition of the state operations, the operation interpreter can be based on different formal languages (e.g., Regular Expressions, SQL).

We use T5v1.1 (Raffel et al., 2020) as the backbone for *Diable*. During training, we use teacher forcing (Goyal et al., 2016) and pass the oracle dialogue state in the input, $\mathcal{B}_{t-1}$. At test time, we pass the previously predicted state, $\hat{\mathcal{B}}_{t-1}$, instead. To learn the model, we optimise the negative log-likelihood of the state operations given the dialogue context, that is

$$\mathcal{L}(\theta)_t = -\log P(\tau_s(\mathcal{O}_t)|\tau_c(C_t)) \quad (3)$$

where f_θ is used to parameterise the probabilistic model P. We use the Adafactor optimiser (Shazeer and Stern, 2018) with no weight decay and we set the learning rate to 10^{-4} with a constant schedule. We fix the training budget at 40k optimiser steps and set the batch size to 32. We generate the output sequence using beam search decoding with 4 beams. We describe in detail the training and inference processes in Appendix C.

3.3 Representing the State as a Table

In our approach, we represent the dialogue state as a table that is sequentially updated. Specifically, a state, $\mathcal{B}$, is represented by a simple two-column table in which the first column is used for the slot name and the second for the slot value (Figure 1). We define the slot name as the concatenation of domain and slot separated by a dash, e.g., `restaurant-area` (see Appendix B). The state table is passed into the dialogue context by simple "linearisation" (Suhr et al., 2020; Scholak et al., 2021; Shaw et al., 2021): the rows are converted to slot-value tuples, cast to a string using the template `{slot} = {value}`, and concatenated together using ; as a separator.[2] During the linearisation, we randomise the order of the rows to avoid overfitting to specific positional biases.

3.4 State Tracking via Table Operations

We introduced how we operationalise table operations as a combination of strings defining the operations and an interpreter that applies the operations to the state. Specifically, in our implementation of *Diable*, we use a simplified formal language consisting of two slot-level operations, `INSERT` and `DELETE`, and a simple regex-based interpreter. The choice of the available operations is motivated by the nature of the MultiWoz datasets which include mostly insertions and deletions. We use the `INSERT` operation also to update the value of slots that are already present in the state. When no operation is needed, the target is defined as the literal string `none`. Updates are less frequent and are caused mostly by inconsistent annotations. In our preliminary experiments, we empirically found that adding an `UPDATE` operation does not improve performance despite adding complexity; thus, we decided to not use it. We emphasise that the specific definition of the operations is not critical for the efficiency of our method and it can be easily adapted to any specific use case. To convert operations to strings we use the template `{command} {slot} = {value}`. If multiple operations need to be applied, we concatenate them using ; as a separator (see Appendix B). We define the target sequence as the concatenation of all the slot-level operations. Since the order in which the operations are applied does not affect the output, we randomise their position during training.

4 Experiments

In this section, we present our experimental setup and provide details about the baselines approaches.

[2] More complex table encoding methods can be applied (Herzig et al., 2020; Yang et al., 2022; Nassar et al., 2022). See the discussion in §6.5.

Model	Architecture	Extra Supervision	Context (C_t)	2.1	2.2	2.4
Cumulative State Models						
TRADE (Wu et al., 2019)	BERT-base (110M)	Schema	$D_{1:t}$	‡45.60	‡45.40	‡55.10
SUMBT (Lee et al., 2019)	BERT-base (110M)	Schema	$D_{1:t}$	‡49.20	‡49.70	‡61.90
DS-DST (Zhang et al., 2020)	BERT-base (110M)	Schema + Pick.	$D_{1:t}$	‡51.21	‡51.70	-
TripPy (Heck et al., 2020)	BERT-base (110M)	-	$D_{1:t}$	‡55.30	†53.52	‡59.60
SAVN (Wang et al., 2020)	BERT-base (110M)	Schema	$D_{1:t}$	‡54.50	-	‡60.10
Seq2Seq-DU (Feng et al., 2021)	2x BERT-base (220M)	Schema + Desc.	$D_{1:t}$	†56.10	‡54.40	-
SimpleTOD (Hosseini-Asl et al., 2020)	GPT-2 (117M)	Schema	$D_{1:t}$	‡50.3/55.7	⋆54.02	-
AG-DST (Tian et al., 2021)	PLATO-2 (310M)	Schema	$D_{t-1:t} + \mathcal{B}_{t-1}$	-	57.26	-
DaP (seq.) (Lee et al., 2021)	T5-base (220M)	Schema + Desc.	$D_{1:t}$	-	51.20	-
DaP (ind.) (Lee et al., 2021)	T5-base (220M)	Schema + Desc.	$D_{1:t}$	56.66	57.60	-
Seq2seq (Zhao et al., 2021)	T5-base (220M)	Pre-training	$D_{1:t}$	⋄52.80	57.60	67.10
*light*Cumulative (Our impl.)	T5v1.1-base (247M)	-	$D_{1:t}$	$53.91_{\pm 0.63}$	$57.01_{\pm 0.45}$	$67.56_{\pm 0.52}$
D3ST (Zhao et al., 2022)	T5-base (220M)	Schema + Pick. + Desc.	$D_{1:t}$	54.20	56.10	72.10
Efficient Models						
MinTL (Lin et al., 2020b)	BART-large (406M)	-	$\mathcal{B}_{t-1}$	53.62	-	-
SOM-DST (Kim et al., 2020)	BERT-base + GRU (113M)	Schema	$D_{t-4:t} + \mathcal{B}_{t-1}$	‡53.68	⋆53.81	‡66.80
Transf.-DST (Zeng and Nie, 2020)	BERT-base (110M)	Schema	$D_{t-4:t} + \mathcal{B}_{t-1}$	**55.35**	-	-
Diable (Ours)	T5v1.1-base (247M)	-	$\mathcal{B}_{t-1}$	⋄$53.91_{\pm 0.70}$	$56.30_{\pm 0.67}$	$70.03_{\pm 0.95}$
	T5v1.1-base (247M)	-	$D_{t-4:t} + \mathcal{B}_{t-1}$	⋄$53.97_{\pm 0.66}$	$\mathbf{56.48}_{\pm 0.57}$	$\mathbf{70.46}_{\pm 1.18}$

Table 1: JGA on the test sets of MultiWoz (2.1, 2.2 and 2.4) for models trained on the respective training sets (note that 2.1 and 2.4 share the same training data). Baseline results reported from the original papers or, when not available, from ⋆: Tian et al. (2021), †: Wang et al. (2022), ‡: Zhao et al. (2021). The column "Context" reports the dialogue context: D and $\mathcal{B}$ denote the dialogue utterances and the set of previous states, respectively. The notation $_{i:j}$ indicates turns from i to j (included). The column "Extra supervision" reports the additional information used (e.g., data augmentation, pre-training, etc.). For Multiwoz 2.1 most baselines use data preprocessing; we denote methods that do not use data preprocessing with ⋄. Underlined the best results overall; **bold** the best results within efficient methods.

4.1 Datasets

The MultiWoz dataset (Budzianowski et al., 2018) is a collection of 10k multi-domain task-oriented human-to-human conversations. It is one of the most used benchmarks in the DST literature (Jacqmin et al., 2022). Nonetheless, it is known to contain annotation errors and previous work proposed different versions (Eric et al., 2020; Han et al., 2021; Ye et al., 2022b) and data normalisation procedures[3] to mitigate this issue. Thus, it is difficult to have a fair comparison of results across the literature. Following the MultiWoz convention (Wu et al., 2019), we filter out dialogues in the "bus", "police", and "hospital" domains (and the respective slots from multi-domain dialogues), and remove the invalid dialogue `SNG01862.json`. We experiment with multiple versions (2.1, 2.2, and 2.4) and use the data as-is (see Appendix B). To construct the training set, we extract the operations automatically from the dataset.

4.2 Evaluation

We use Joint Goal Accuracy (Henderson et al., 2014, JGA) as the main metric for all experiments: it measures the proportion of turns for which the predicted state (slot-value pairs) exactly match the gold label. At each turn, for each slot, a list of acceptable values is included in the annotation (e.g., `hotel-name: ["marriott", "marriott hotel"]`). We consider a value correct if it exactly matches one of the available options. Importantly, we perform an uncased evaluation since the annotation casing is not consistent.

4.3 Cumulative State and Efficient Baselines

We compare our results with a set of strong cumulative state models (i.e., models that use all previous turns and output a value for each slot at each turn, see Figure 2), and efficient baseline models. We also implement our own version of a cumulative state model and its "lighter" variant, *light*Cumulative: the state does not include the inactive slots. In all our experiments, the full cumulative models underperform *light*Cumulative while being less efficient (∼1.18x slower). Thus, we only report the results of *light*Cumulative, effectively selecting a stronger baseline.

In the upper part of Table 1 ("Cumulative State Models"), we include results from state-of-the-art generative cumulative state models. In each section,

[3]Most notably the TRADE scripts from Wu et al. (2019) to normalise both text and labels.

we report details and results for encoder-based, sequence-to-sequence, and T5-based models, respectively. The latter class of models based on T5 is related to our implementation of *Diable* in that they share the same backbone. However, they are not directly comparable due to the additional text preprocessing and label normalisation. The results of our own re-implementation of a cumulative state model, *light*Cumulative, are directly comparable as we adopted the same experimental conditions.

In the bottom part of Table 1 ("Efficient Models"), we report the JGA of the latest generative efficient DST approaches in the literature. Despite being related to our implementation of *Diable*, these approaches are not directly comparable since they rely on additional information (e.g. the schema) or are based on a different backbone model.

Context	Runtime (ms)	Speedup (↑)
Cumulative		
$D_{1:t}$	40.83	1.00
***light*Cumulative**		
$D_{1:t} + \mathcal{B}_{t-1}$	34.28	1.19
$D_{1:t}$	33.65	1.21
$D_{t-4:t} + \mathcal{B}_{t-1}$	32.56	1.25
$\mathcal{B}_{t-1}$	30.62	1.33
Diable		
$D_{1:t} + \mathcal{B}_{t-1}$	19.59	2.09
$D_{1:t}$	19.41	2.11
$D_{t-4:t} + \mathcal{B}_{t-1}$	18.29	2.23
$\mathcal{B}_{t-1}$	17.17	2.38

Table 2: Median instance-level runtime in milliseconds and relative speed-up *vs* a cumulative state baseline.

5 Results

In this section, we discuss our experimental results. In Table 1, we summarise the JGA on three versions of MultiWoz (2.1, 2.2, and 2.4) for both *Diable* and the baseline models. The results for the baseline models are taken from previous work (Tian et al., 2021; Wang et al., 2022; Zhao et al., 2021) when better or if missing for a particular version in the original papers. The results for *Diable* and *light*Cumulative implementation are averaged across 5 random seeds.

5.1 *Diable* and Cumulative State Models

We compare *Diable*'s performance to cumulative state models, i.e., models that have access to all previous turns in the conversation. We emphasise that *Diable* uses none or a limited number of previous turns and thus has less context with respect to these models. On one hand, our goal is to evaluate the trade-off between efficiency and performance; on the other hand, to study the capability of the system to generate correct table operations.

The cumulative state model results are shown in the first part of Table 1. First, D3ST (Zhao et al., 2022) achieves the best JGA score on MultiWoz 2.4 when the backbone is T5-base. Similarly to *Diable*, D3ST is based on a T5 model; however, it has access to more information such as the schema, the slot descriptions, and the list of possible values ("picklist") for categorical slots. Nonetheless, *Diable* scores within 1 standard deviation in terms of JGA, while being more than 2x more efficient.

When the backbone model of D3ST (Zhao et al., 2022) is T5-xxl (11B), it scores 57.80, 58.70, and 75.90, respectively, on the three versions of the MultiWoz dataset. These scores are significantly higher than all other baselines. However, this improvement is solely due to increasing the model size, and we argue that the same performance improvement can be achieved by scaling the backbone of *Diable* to larger models. In particular, error analysis shows that most of the errors in our instantiation of *Diable*-based systems are due to the model not recognising active slots ("under-predictions"). A larger backbone model can alleviate this issue by picking up less obvious contextual cues. Finally, the difference is more significant for version 2.1 because D3ST also applies text preprocessing, as used in other baselines. Moreover, baselines that use smaller models (the first part of Table 1) consistently score lower than those based on the larger and better pre-trained T5 checkpoints. The only exception is AG-DST (Tian et al., 2021) but their backbone model has 310M parameters.

We further compare *Diable* to our implementation of a cumulative state T5-based model (*light*Cumulative). This comparison is fairer, as the models are put in the exact same experimental conditions. Our goal here is to quantify the improvements due to our proposed approach isolating additional effects from model pre-training and architectural changes. The results show that our *Diable* approach has a significantly better JGA (+3 absolute points) on the less noisy version of MultiWoz (i.e., 2.4) and has similar performance on 2.1 and 2.2, while still being more efficient.

Next, we compare *Diable* with two other strong models based on the T5 architecture (making them directly comparable, besides the preprocessing steps): *DaP (ind)* (Lee et al., 2021) and *Seq2seq* (Zhao et al., 2021). Both models achieve a slightly higher JGA than *Diable* on 2.2 (1 point absolute); however, they are again less efficient and have access to a larger context. *DaP* relies on slot descriptions (thus, the schema) and runs inference once for every slot, which is not scalable to large schemas. The improvements in *Seq2seq* are likely due to its extensive DST-specific pre-training.

Our results confirm that *Diable*-based systems while being efficient, achieve competitive JGA on MultiWoz as compared to both other strong efficient DST baselines and cumulative state state-of-the-art DST systems without requiring any ad-hoc data preprocessing, access to the full history, extra supervision, or large backbone models.

5.2 *Diable* and Efficient Models

Comparing *Diable* to other efficient state-of-the-art DST models, that are based on state operations, we see significant improvements up to almost 4 JGA points on version 2.4 (shown in the "Efficient Models" section of Table 1). Only Transformer-DST (Zeng and Nie, 2020) is able to outperform our model on 2.1. However, they use data preprocessing (text and label normalisation) and extra supervision (schema). This model is an improved version of SOM-DST (Kim et al., 2020), therefore the same argument applies to the latter, which achieves slightly lower performance even using the same extra supervision and text normalisation.

5.3 Latency Analysis

Table 2 reports the median[4] inference time and the speed-up factor of *Diable* relative to *light*Cumulative. Our approach is more than 2x faster, even when using the full history as context. These results clearly show that the biggest efficiency gains are obtained by shortening the output sequence, that is, replacing the cumulative state with state operations. Consequently, adding only the last N turns comes at a small cost for a *Diable* while potentially helping the model to recover values not present in the current dialogue context. Using the Inference Time Complexity (ITC) notation introduced by Ren et al. (2019), our proposed approach has $\Omega(1)$ and $O(N)$, where N is the number of slots in the schema, as the best and worst case ITC, respectively. Whereas, SOM-DST and Transformer-DST have a best-case ITC of $\Omega(N)$.

[4]We report the median as the distribution of the inference time is left-skewed.

Train → Test	***light*Cumulative**	***Diable***
Context: $D_{1:t}$		
$2.2 \rightarrow 2.2$	$57.01_{\pm 0.45}$	$55.63_{\pm 0.68}$
$2.2 \rightarrow 2.4$	$63.11_{\pm 0.83}$	$64.95_{\pm 0.55}$
Context: $\mathcal{B}_{t-1}$		
$2.2 \rightarrow 2.2$	$56.50_{\pm 0.47}$	$56.30_{\pm 0.67}$
$2.2 \rightarrow 2.4$	$63.52_{\pm 0.96}$	$66.13_{\pm 0.97}$

Table 3: Effect on JGA (mean ± 1 standard deviation) of different context and state representations.

5.4 Robustness to Noisy Annotations

Table 3 compares the performance of models trained on MultiWoz 2.2 with different context and state representations. Notably, when evaluated on the cleaner 2.4 version (bottom row for both parts of the table), *Diable* consistently outperforms *light*Cumulative. In fact, regardless of the dialogue context, *Diable* achieves a better JGA on 2.4. We hypothesise that the lower accuracy of *light*Cumulative is due to overfitting the noisy annotations of the training set. In particular, we think that since it generates the full state from scratch at every turn, the decoder might learn wrong correlations amongst slots that are wrongly annotated in the training set. For example, `hotel-type` and `attraction-type` are inconsistently and sparsely annotated in the training set, while in the test set of version 2.4 they tend to appear almost always together with the respective `hotel-name` and `attraction-name` slots. Thus, a cumulative state model can learn to not generate one when the other is present. Instead, being *Diable* based on state *changes*, we presume that it learns to treat slots more independently.

6 Discussion

Our task formalisation is intuitively simple and is especially beneficial for large pre-trained sequence-to-sequence models. First, the state is expanded sequentially and thus only includes the necessary slots. This minimises the size of the input context, allowing the models to scale to larger schemas before reaching their maximum input length. Second, since the model needs to focus on the state *changes*, the decoder only needs to generate operations for a

Context	Test Set	JGA	
Cumulative state		base	large
$D_{1:t}$	2.2	57.37	57.01
$D_{1:t}$	2.4	65.82	63.11
Diable		base	large
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2	56.74	56.48
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4	65.01	65.35

Table 4: MultiWoz 2.2 and 2.4 test set JGA for T5v1.1 `base` and `large` trained on the MultiWoz 2.2.

Train → Test	Predicted	Gold
2.1 → 2.1	$53.91_{\pm 0.70}$	$80.65_{\pm 0.24}$
2.1 → 2.4	$70.03_{\pm 0.95}$	$90.14_{\pm 0.30}$
2.2 → 2.2	$56.30_{\pm 0.67}$	$82.50_{\pm 0.28}$
2.2 → 2.4	$66.13_{\pm 0.97}$	$88.29_{\pm 0.35}$

Table 5: JGA (mean ±1 standard deviation) with gold and the predicted previous state in the input context.

limited number of slots (previous slots persist implicitly in the state, no need for explicit "carryover" operations). Third, our system is general in that it deals with span-based and categorical slots in the same way, and outputs both the operations and the slot-value pairs in a single forward pass, without the need for specialised architectures. Finally, since not all pre-defined slots are needed in the input, we do not have to access the schema beforehand, and thus it can be learned from the data directly.

6.1 Impact of the Dialogue History

Table 3 compares the effect of the context size for both *light*Cumulative and *Diable* trained on version 2.2. Comparing the results from the upper and bottom parts of the table, we see that using only the previous state barely changes the JGA of *light*Cumulative but benefits *Diable*. We hypothesise that being a cleaner and more compact representation of the conversation, the previous state introduces less noise than the full history. This is especially true in conversations for which the value of a slot is changed or removed throughout the conversation. However, completely removing the dialogue history reduces the ability of the model to recover values referenced at the beginning of the conversation. We hypothesise that this negative effect is not too evident because of the entity bias present in the MultiWoz dataset (Qian et al., 2021) that allows the model to memorise and correctly predict values for certain slots even when not present in the dialogue context (§6.4). Finally, when evaluated on the cleaned version 2.4, *Diable* consistently matches or outperforms *light*Cumulative.

6.2 Impact of the Model Size

Table 4 compares the performance of the base and large version of T5v1.1 for both *light*Cumulative and *Diable* models. We find that scaling up model size does not improve JGA, however, we hypothesise that scaling it further can improve the performance similarly to D3ST (Zhao et al., 2022).

6.3 Impact of the State Representation

When replacing the tabular state representation with a cumulative one in *Diable*, *ceteris paribus*, we find a 3% reduction in JGA for version 2.4 and up to 5% for other versions. Specifically, at the beginning of the conversation, the state includes all the slots with the `none` value. In this case, the `INSERT` operation is unchanged while the `DELETE` operation becomes an update with a `none` value.

6.4 Error Propagation

Diable, like any recursive state model (Zhao et al., 2021), is affected by error propagation: since we pass the previous predicted state at each turn, errors can be persisted. We measure the potential gains stemming from completely avoiding the error propagation by using the *gold* previous state rather than the predicted one in the dialogue context. Table 5 reports the upper bound on JGA for our simple *Diable* instantiation and highlights that there is potential to improve JGA by adopting recent methodologies targeted at reducing error propagation (Zhang et al., 2022).

In our experiments, we identify two main sources of error propagation that account for more than 60% of the total mistakes: state "under-prediction" (i.e., the model does not recognise that a certain state is active) and value misprediction. Under-prediction happens when the system is unable to recognise that specific slots are active. Since MultiWoz presents a strong entity bias—e.g., "Cambridge" appears in 50% of the destination cities in the training data (Qian et al., 2021)—a possible direction to address this issue is to use data augmentation methods targeted at reducing entity bias and annotation inconsistency (Summerville et al., 2020; Lai et al., 2022) by improving the overall slot recall. Value misprediction happens when

the value for a correctly predicted slot is wrong. This is especially evident when the same slot is discussed in multiple turns and its value can potentially change. One way to address this limitation is by automatically pre-selecting the previous dialogue turns to include the relevant information about a specific slot in the context window (Yang et al., 2021; Guo et al., 2022; Wang et al., 2022).

We do not constrain the generation in any way, and thus *Diable* can generate invalid slots or values (e.g., `attraction-time`). In our experiments, errors due to invalid states are rare (less than 2% of the total mistakes): in fact, using the schema to filter incorrectly predicted slots at each turn did not improve the JGA significantly (less than 1%). There are several promising techniques that can further improve the performance of our system, at a minor efficiency cost, such as amendable generation (Tian et al., 2021), constrained generation (Lin et al., 2020a), and schema descriptions (Lee et al., 2021; Zhao et al., 2022). Finally, with larger schemas and more diverse conversations, constraining the set of values that the model can predict can potentially further improve performance and safety.

6.5 Future Directions

In §5, we showed that *Diable* is an effective DST approach, at the same time it is competitive with budget-matched (in terms of parameter count) cumulative state baselines. We emphasise that our goal is not to reach state-of-the-art JGA on the MultiWoz dataset. We intentionally keep our *Diable*-based models as simple as possible, by not adding extra supervision signals, to clearly measure the effectiveness of our approach. However, the benefits coming from *Diable* can be easily added on top of other methods. We believe our approach can be improved and expanded in several ways.

Explicitly Modelling Slot Dependence. *Diable* treats slots independently of each other and implicitly uses the model's capability of learning their co-occurrence patterns. However, as the schema becomes larger and the dialogues longer, slot dependence becomes more complex and the model might fail to learn it effectively. Explicitly modelling the slot dependence can potentially improve performance, robustness (to spurious correlations), and efficiency. For example, selecting only relevant turns from the dialogue history as context to predict slot values. In our experiments, we show consistent improvement across all MultiWoz versions by adding the previous 4 dialogue turns in the dialogue context (Table 1 – last 2 rows). However, this simple heuristic might be suboptimal when the schema is large and the dialogue is long because relevant turns may not be the immediately preceding ones and we might add irrelevant context or omit relevant information. Instead, adopting a more granular turn selection method based on the slot dependence (Yang et al., 2021; Guo et al., 2022) can improve both performance and efficiency.

Improving Table Representations. When passing the previous state in the context, we simply linearise the table. That is, we represent the previous states as discrete tokens passed in the input context for the next turn. This allowed us to use the T5 architecture without modification. A promising direction for future work is to use continuous representations for the state table (Wu et al., 2022). This representation can potentially require fewer or no tokens to represent the state, thus further improving the efficiency of our approach.

7 Conclusions

In this paper, we introduce a novel efficient formulation of the DST task and a new system, *Diable*, specifically designed to enhance efficiency and optimise the capabilities of state-of-the-art sequence-to-sequence models. *Diable* represents the dialogue state as an implicit table and updates it using a sequence-to-sequence model to generate table operations at each turn. Our task formalisation provides a significant efficiency gain (up to 2.4x speed-up in inference time) compared to cumulative state approaches adopted by the current state-of-the-art DST systems. Moreover, this sizeable improvement comes with a minimal efficiency-accuracy trade-off. In fact, *Diable* outperforms other efficient DST approaches in the literature by more than 3 absolute JGA points on MultiWoz 2.4 and shows a competitive performance with respect to current DST state-of-the-art systems. *Diable* comes with other advantages: it is simple and general (it makes no assumptions about the schema and does not require any specialised architecture) and it is robust to noise. Moreover, it allows to plug and play sequence-to-sequence models without any architectural modification easily. Finally, our approach goes beyond the dialogue setting and can be adapted to the sequential processing of long documents for information extraction tasks with memory-constrained language models.

Acknowledgements

We thank the anonymous reviewers for their helpful questions and comments, which have helped us improve the quality of the paper. We sincerely thank Miguel Ballesteros, Yogarshi Vyas, Neha Anna John, Yi Zhang, Paula Czarnowska, Laura Aina, Thomas Mueller, and Marcello Federico for their constructive and detailed feedback on the early versions of the paper. We thank Tiago Pimentel, Josef Valvoda, Davide Lesci, and Marco Lesci for their feedback on the final version.

Limitations

In Section 6.4, we already discussed the limitations and challenges of the model proposed (e.g., the model has access to less contextual information from the conversation history, errors can propagate more easily as it does not re-predict the entire cumulative state at each step, and mistakes could only be fixed by explicit delete or update operation). In the following, we concentrate on the limitations that refer to the scope of this work.

Languages. We experimented with a limited number of languages (English) and datasets (MultiWoz 2.1, 2.2 and 2.4). We do not have experimental evidence that our method can work for other languages, including languages with a richer morphology. Still, our system has been built without any language-specific constraints or resources, other than the T5 checkpoints and the manually annotated training set. Our method can be applied to any other language (without modification) for which these resources are available, or by applying cross-lingual techniques and resources (e.g., multilingual language models, translation/projection of the training set) to transfer to other languages zero-shot. In those cases, the expected quality is lower, but the efficiency advantage of *Diable* remains.

Models. We experimented with two models (T5v1.1 base and large). This is due to the restriction on our computational budget to be both economically- and environmentally-friendly, which made it infeasible to conduct thorough experiments using larger-scale language models. However, we re-emphasise that *Diable* allows one to easily plug and play arbitrary language models and the efficiency advantage of *Diable* remains.

Diversity in the Evaluation Dataset. We experimented with three different versions of the MultiWoz dataset (2.1, 2.2, and 2.4). Although this is the current benchmark for DST accepted by the community, and we followed the standard evaluation methodology and metrics, we are aware that the results presented might not be directly generalisable to other datasets or real-world scenarios with a considerable data shift with respect to MultiWoz. Additionally, MultiWoz has a certain level of noise and this can have an impact on the evaluation and the generalisation capabilities of the model trained.

References

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. In *Advances in neural information processing systems*, volume 33, pages 1877–1901.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. 2018. MultiWOZ – A large-scale multi-domain Wizard-of-Oz dataset for task-oriented dialogue modelling. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5016–5026, Brussels, Belgium.

Guan-Lin Chao and Ian R. Lane. 2019. BERT-DST: Scalable end-to-end dialogue state tracking with bidirectional encoder representations from transformer. In *20th Annual Conference of the International Speech Communication Association*, Interspeech 2019, pages 1468–1472, Graz, Austria.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, Albert Webson, Shixiang Shane Gu, Zhuyun Dai, Mirac Suzgun, Xinyun Chen, Aakanksha Chowdhery, Alex Castro-Ros, Marie Pellat, Kevin Robinson, Dasha Valter, Sharan Narang, Gaurav Mishra, Adams Yu, Vincent Zhao, Yanping Huang, Andrew Dai, Hongkun Yu, Slav Petrov, Ed H. Chi, Jeff Dean, Jacob Devlin, Adam Roberts, Denny Zhou, Quoc V. Le, and Jason Wei. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, Adarsh Kumar, Anuj Goyal, Peter Ku, and Dilek Hakkani-Tur. 2020. MultiWOZ 2.1: A consolidated multi-domain dialogue

dataset with state corrections and state tracking baselines. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 422–428, Marseille, France.

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*, pages 1714–1725, Online.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *Proceedings of the 20th Annual SIGdial Meeting on Discourse and Dialogue*, pages 264–273, Stockholm, Sweden.

Anirudh Goyal, Alex Lamb, Ying Zhang, Saizheng Zhang, Aaron Courville, and Yoshua Bengio. 2016. Professor forcing: A new algorithm for training recurrent networks. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NIPS'16, page 4608–4616, Red Hook, NY, USA.

Jinyu Guo, Kai Shuang, Jijie Li, Zihan Wang, and Yixuan Liu. 2022. Beyond the granularity: Multi-perspective dialogue collaborative selection for dialogue state tracking. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 2320–2332, Dublin, Ireland.

Ting Han, Ximing Liu, Ryuichi Takanabu, Yixin Lian, Chongxuan Huang, Dazhen Wan, Wei Peng, and Minlie Huang. 2021. MultiWOZ 2.3: A multi-domain task-oriented dialogue dataset enhanced with annotation corrections and co-reference annotation. In *Natural Language Processing and Chinese Computing: 10th CCF International Conference*, NLPCC '21, page 206–218, Qingdao, China.

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, Virtual.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014. Word-based dialog state tracking with recurrent neural networks. In *Proceedings of the 15th Annual Meeting of the Special Interest Group on Discourse and Dialogue (SIGDIAL)*, pages 292–299, Philadelphia, PA, U.S.A.

Jonathan Herzig, Pawel Krzysztof Nowak, Thomas Müller, Francesco Piccinno, and Julian Eisenschlos. 2020. TaPas: Weakly supervised table parsing via pre-training. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4320–4333, Online.

Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. In *Advances in Neural Information Processing Systems*, volume 33, pages 20179–20191, Online.

Léo Jacqmin, Lina M. Rojas Barahona, and Benoit Favre. 2022. "Do you follow me?": A survey of recent approaches in dialogue state tracking. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 336–350, Edinburgh, UK.

Sungdong Kim, Sohee Yang, Gyuwan Kim, and Sang-Woo Lee. 2020. Efficient dialogue state tracking by selectively overwriting memory. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 567–582, Online.

Taku Kudo and John Richardson. 2018. SentencePiece: A simple and language independent subword tokenizer and detokenizer for neural text processing. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 66–71, Brussels, Belgium.

Adarsh Kumar, Peter Ku, Angeliki Metallinou, Anuj Goyal, and Dilek Hakkani-Tür. 2020. MA-DST: Multi-attention-based scalable dialog state tracking. In *AAAI 2020*, Online.

Chun-Mao Lai, Ming-Hao Hsu, Chao-Wei Huang, and Yun-Nung Chen. 2022. Controllable user dialogue act augmentation for dialogue state tracking. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 53–61, Edinburgh, UK.

Chia-Hsuan Lee, Hao Cheng, and Mari Ostendorf. 2021. Dialogue state tracking with a language model using schema-driven prompting. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 4937–4949, Online and Punta Cana, Dominican Republic.

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

Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. 2018. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1437–1447, Melbourne, Australia.

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*, pages 7871–7880, Online.

Lizi Liao, Le Hong Long, Yunshan Ma, Wenqiang Lei, and Tat-Seng Chua. 2021. Dialogue state tracking with incremental reasoning. *Transactions of the Association for Computational Linguistics*, 9:557–569.

Bill Yuchen Lin, Wangchunshu Zhou, Ming Shen, Pei Zhou, Chandra Bhagavatula, Yejin Choi, and Xiang Ren. 2020a. CommonGen: A constrained text generation challenge for generative commonsense reasoning. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 1823–1840, Online.

Zhaojiang Lin, Andrea Madotto, Genta Indra Winata, and Pascale Fung. 2020b. MinTL: Minimalist transfer learning for task-oriented dialogue systems. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 3391–3405, Online.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *7th International Conference on Learning Representations*, ICLR '19, New Orleans, Louisiana, USA.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, pages 794–799, Beijing, China.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, pages 1777–1788, Vancouver, Canada.

Ahmed Nassar, Nikolaos Livathinos, Maksym Lysak, and Peter Staar. 2022. TableFormer: Table structure understanding with transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, CVPR '22, pages 4614–4623, New Orleans, Louisiana, USA.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. PyTorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, volume 32, Red Hook, New York, USA.

Kun Qian, Ahmad Beirami, Zhouhan Lin, Ankita De, Alborz Geramifard, Zhou Yu, and Chinnadhurai Sankar. 2021. Annotation inconsistency and entity bias in MultiWOZ. In *Proceedings of the 22nd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 326–337, Singapore and Online.

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. *Journal of Machine Learning Research*, 21(140):1–67.

Abhinav Rastogi, Dilek Hakkani-Tur, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *IEEE Workshop on Automatic Speech Recognition and Understanding*.

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 *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*, pages 8689–8696, New York, NY, USA.

Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. 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 1876–1885, Hong Kong, China.

Torsten Scholak, Nathan Schucher, and Dzmitry Bahdanau. 2021. PICARD: Parsing incrementally for constrained auto-regressive decoding from language models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 9895–9901, Online and Punta Cana, Dominican Republic.

Peter Shaw, Ming-Wei Chang, Panupong Pasupat, and Kristina Toutanova. 2021. Compositional generalization and natural language variation: Can a semantic parsing approach handle both? In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*, pages 922–938, Online.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive learning rates with sublinear memory cost. In *Proceedings of the 35th International Conference on Machine Learning*, volume 80 of *Proceedings of Machine Learning Research*, pages 4596–4604, Stockholm, Sweden.

Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 8372–8388, Online.

Adam Summerville, Jordan Hashemi, James Ryan, and William Ferguson. 2020. How to tame your data: Data augmentation for dialog state tracking. In *Proceedings of the 2nd Workshop on Natural Language Processing for Conversational AI*, pages 32–37, Online.

Xin Tian, Liankai Huang, Yingzhan Lin, Siqi Bao, Huang He, Yunyi Yang, Hua Wu, Fan Wang, and Shuqi Sun. 2021. Amendable generation for dialogue state tracking. In *Proceedings of the 3rd Workshop on Natural Language Processing for Conversational AI*, pages 80–92, Online.

Yexiang Wang, Yi Guo, and Siqi Zhu. 2020. Slot attention with value normalization for multi-domain dialogue state tracking. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*, EMNLP '20, pages 3019–3028, Online.

Yifan Wang, Jing Zhao, Junwei Bao, Chaoqun Duan, Youzheng Wu, and Xiaodong He. 2022. LUNA: Learning slot-turn alignment for dialogue state tracking. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3319–3328, Seattle, United States.

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

Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable multi-domain state generator for task-oriented dialogue systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 808–819, Florence, Italy.

Qingyang Wu, Zhenzhong Lan, Kun Qian, Jing Gu, Alborz Geramifard, and Zhou Yu. 2022. Memformer: A memory-augmented transformer for sequence modeling. In *Findings of the Association for Computational Linguistics: AACL-IJCNLP 2022*, pages 308–318, Online only.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1448–1457, Melbourne, Australia.

Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust transformer modeling for table-text encoding. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics*, pages 528–537, Dublin, Ireland.

Puhai Yang, Heyan Huang, and Xian-Ling Mao. 2021. Comprehensive study: How the context information of different granularity affects 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*, pages 2481–2491, Online.

Fanghua Ye, Yue Feng, and Emine Yilmaz. 2022a. ASSIST: Towards label noise-robust dialogue state tracking. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 2719–2731, Dublin, Ireland.

Fanghua Ye, Jarana Manotumruksa, and Emine Yilmaz. 2022b. MultiWOZ 2.4: A multi-domain task-oriented dialogue dataset with essential annotation corrections to improve state tracking evaluation. In *Proceedings of the 23rd Annual Meeting of the Special Interest Group on Discourse and Dialogue*, pages 351–360, Edinburgh, UK.

Fanghua Ye, Jarana Manotumruksa, Qiang Zhang, Shenghui Li, and Emine Yilmaz. 2021. Slot self-attentive dialogue state tracking. In *Proceedings of the Web Conference 2021*, WWW '21, page 1598–1608, Ljubljana, Slovenia.

Pengcheng Yin, Graham Neubig, Wen-tau Yih, and Sebastian Riedel. 2020. TaBERT: Pretraining for joint understanding of textual and tabular data. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8413–8426, Online.

Yan Zeng and Jian-Yun Nie. 2020. Jointly optimizing state operation prediction and value generation for dialogue state tracking. *arXiv preprint arXiv:2010.14061*.

Haoning Zhang, Junwei Bao, Haipeng Sun, Youzheng Wu, Wenye Li, Shuguang Cui, and Xiaodong He. 2022. MoNET: Tackle state momentum via noise-enhanced training for dialogue state tracking. *arXiv preprint arXiv:2211.05503*.

Jianguo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wang, Philip Yu, Richard Socher, and Caiming Xiong. 2020. Find or classify? Dual strategy for slot-value predictions on multi-domain dialog state tracking. In *Proceedings of the Ninth Joint Conference on Lexical and Computational Semantics*, pages 154–167, Barcelona, Spain (Online).

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. *arXiv preprint arXiv:2201.08904*.

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

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 1458–1467, Melbourne, Australia.

Su Zhu, Jieyu Li, Lu Chen, and Kai Yu. 2020. Efficient context and schema fusion networks for multi-domain dialogue state tracking. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 766–781, Online.

Appendix for "*Diable*: Efficient Dialogue State Tracking as Operations on Tables"

A Data Statistics

In this section, we report statistics about versions 2.1-2.4 of the MultiWoz dataset. Table 6 shows the distribution of domains across dialogues and turns. Table 9 reports the distribution of slots across dialogues and turns. Finally, Table 10 reports general statistics regarding the frequency of domains, slots, and turns.

Domain	2.1	2.2	2.3	2.4
Number of dialogues				
attraction	3494	3484	3503	3486
hotel	4190	4182	4228	4188
restaurant	4748	4728	4765	4732
taxi	1879	1872	1884	1875
train	3940	3931	3945	3936
Number of turns				
attraction	24016	23940	24144	23986
hotel	31416	31378	31735	31399
restaurant	33201	33104	33369	33121
taxi	7760	7708	7833	7739
train	27738	27699	27830	27721

Table 6: Frequency of domains across dialogues and turns for MultiWoz 2.1-2.4.

B Data Preprocessing

Data Cleaning. Following the MultiWoz convention, we filter out dialogues in the "bus", "police" and "hospital" domains (and the respective slots from multi-domain dialogues), and we remove the invalid dialogue `SNG01862.json`. The processed dataset contains 5 domains ("restaurant", "train", "hotel", "taxi", "attraction"), 30 slots, 9,917 dialogues, and 79,793 turns.

Slot-Value Representation. We represent states as a list of triplets in the form of `(domain, slot, value)`. We define a slot as the concatenation of the domain name and slot name, e.g., `(restaurant-area, center)`. Annotations can possibly contain multiple acceptable values. During testing, this is not problematic as we consider a prediction correct when the predicted value is contained in the acceptable set of values. However, during training, we need to choose one in order to use teacher forcing. To do so, we first check which one of the possible values is actually a span from the text. If none are present, we choose the longest. Since the casing is inconsistent, we lowercase all the values.

Label Normalisation. MultiWoz contains noisy annotations and different authors have tried to alleviate the issue by devising different label normalisation procedures. For example, the scripts by Wu et al. (2019)[5] and Hosseini-Asl et al. (2020).[6] In this work, we try to balance being as faithful as possible to the original annotations without needlessly penalizing the evaluation of our system. In detail, we target the following noisy annotation values:

- Typos: "guest house", "swimming pool", "night club", "concert hall" that appear with and without spaces. When one is present, we also add the other version as a possible correct answer. This normalisation affects "hotel-type", "attraction-type", "hotel-name", and "attraction-name" slots.
- Spelling inconsistencies: "theater" and "center" appear both in their UK and US English versions. When one is present, we add the alternative version. This normalisation affects the "hotel-area", "restaurant-area", "attraction-area", and "attraction-type".
- Names starting with "the": some names appear with the "the" proposition. In such cases, we add a version without the proposition. This normalisation affects the "hotel-name","restaurant-name", and "attraction-name" slots.
- Categorical slots: the "hotel-star" slot is a categorical slot whose values are integers in 0-8. In some cases, the annotation includes the literal "star" string. In such cases, we remove the "star" from the annotation.

Overall, these are minimal changes. Many such errors caused by noisy labels are still present in the dataset. We leave as future work the creation of an even cleaner evaluation dataset. More details on the impact of these normalisations are available in Appendix D.

Input Creation. In a preliminary study, we experimented with different possible templates for the input sequence and found that, after a certain degree, adding more text to the prompt was not beneficial and the exact wording was not having a

[5] https://github.com/jasonwu0731/trade-dst/blob/master/utils/fix_label.py.

[6] https://github.com/salesforce/simpletod/tree/master/noisy_annotations.

big impact. Therefore, to balance simplicity and accuracy, for all experiments, we used the following simple templates,

- *light*Cumulative:

```
generate full state: dialogue: system: {system_utterance} user: {user_utterance} <sep> previous dialogue states: {previous_states} <sep> history: {history} <sep> states:
```

- *Diable*

```
generate update operations: dialogue: system: {system_utterance} user: {user_utterance} <sep> previous dialogue states: {previous_states} <sep> history: {history} <sep> operations:
```

In our experiments, up to decimal differences, including the schema in the input context does not affect performance. Similarly, there is no impact from excluding the <sep> token to separate the various parts of the input context. However, excluding the prefixes—i.e., `generate update operations` and `generate full state`—reduces performances by up to 1.2% JGA for both state operations and cumulative state models. A similar effect is caused by removing the "system"/"user" identifiers (as also observed by (Hosseini-Asl et al., 2020).

Note that we did not optimise over the choice of the prefixes; given the new instruction fine-tuned models recently proposed (Chung et al., 2022), we hypothesise—and leave for future work—that different prompts can improve the DST, especially in the few-shot setting. Finally, lower-casing the text decreases performances by up to 1% JGA.

An example of an actual processed conversation is shown in Table 7.

C Training Details

Hardware Details. We used a server with 8 Tesla V100 GPUs. The batch size on each GPU was limited to 8. Thus, we ran the majority of the experiments in a multi-GPU setting with 4 GPUs allocated to each training job with no need for gradient accumulation. *Diable* models were trained in 3-4 hours, while cumulative state models required 5-6 hours.

Below, we report the output of the `lscpu` command (excluding the flags):

```
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                64
On-line CPU(s) list:   0-63
Thread(s) per core:    2
Core(s) per socket:    16
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 79
Model name:            Intel(R) Xeon(R)
                        CPU E5-2686 v4
                        @ 2.30GHz
Stepping:              1
CPU MHz:               2700.216
CPU max MHz:           3000.0000
CPU min MHz:           1200.0000
BogoMIPS:              4600.04
Hypervisor vendor:     Xen
Virtualization type:   full
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              46080K
NUMA node0 CPU(s):     0-15,32-47
NUMA node1 CPU(s):     16-31,48-63
```

Model. For all experiments, we use the T5 architecture (Raffel et al., 2020) and use the associated T5v1.1 base[7] and large[8] checkpoints available on the HuggingFace Hub via the transformers (Wolf et al., 2020) library and implemented using the PyTorch (Paszke et al., 2019) framework. In a preliminary study, we compared T5v1.1 with the original T5 and flan-T5 (Chung et al., 2022) variants and did not see any significant differences; we choose to use the T5v1.1 checkpoint since it is not fine-tuned on downstream tasks.

Data Preparation. We use the default SentencePiece (Kudo and Richardson, 2018) tokenizer with vocabulary size 32k associated with the T5v1.1 checkpoint and available in the tokenizers library (Wolf et al., 2020). We add <sep> to the vocabulary as a special separator token. We truncate only the input sequence at 512 tokens during training but do not truncate during evaluation in order to not penalize cumulative state models (our main baseline).

Training. We use the Pytorch-Lightning[9] library to implement the training loop. For all experiments, we use the Adafactor optimiser (Shazeer and Stern,

[7] https://huggingface.co/google/t5-v1_1-base.
[8] https://huggingface.co/google/t5-v1_1-large.
[9] https://github.com/Lightning-AI/lightning/.

Turn	Utterances		State	Processed data	
	System	User		Input	Target
0	none	I'm looking for a place to stay. It needs to be a guesthouse and include free wifi.	{'hotel-internet': 'yes', 'hotel-type': 'guesthouse'}	generate update operations: dialogue: system: none user: I'm looking for a place to stay. It needs to be a guesthouse and include free wifi. <sep> previous dialogue states: none <sep> operations:	INSERT hotel-internet = yes ; INSERT hotel-type = guesthouse
1	There are 23 hotels that meet your needs. Would you like to narrow your search by area and/or price range?	I would like for it to be cheap and include free parking.	{'hotel-internet': 'yes', 'hotel-parking': 'yes', 'hotel-pricerange': 'cheap', 'hotel-type': 'guesthouse'}	generate update operations: dialogue: system: There are 23 hotels that meet your needs. Would you like to narrow your search by area and/or price range? user: I would like for it to be cheap and include free parking. <sep> previous dialogue states: hotel-type = guesthouse; hotel-internet = yes <sep> operations:	INSERT hotel-parking = yes ; INSERT hotel-pricerange = cheap

Table 7: Example of a conversation from the MultiWoz dataset (dialogue `MUL0003.json`) processed according to our task formalization. The "Input" column shows the template used to construct the input sequence. It includes optional fields separated by the <sep> token. For example, if the context also includes the dialogue history, we add `<sep> history: {history}` before `operations`. In red the states from the previous turn. We use the value none to fill-in empty utterances (e.g., the first system utterance), states (e.g., the first state is always empty), or no-operations (i.e., when the state does not need to be updated.

2018) with no weight decay, `eps=[1e-30, 0.001]`, `clip_threshold: 1.0`, `decay_rate: -0.8`, `beta1: null`, `scale_parameter: false`, `relative_step: false`, and `warmup_init: false`. We set the learning rate to 10^{-4} and use a constant schedule. We fix the training budget at 40k optimiser steps and set the batch size to 32 to trade off the speed and precision of the gradient estimation.

Inference. For all experiments, we use beam search decoding, as implemented in the HuggingFace library (Wolf et al., 2020) with 4 beams and no additional settings.

Reproducibility. For reproducibility, we use the pseudo-random seed for both data shuffling and model initialization. In each experiment, we fix the seed for pseudo-random number generators and use CUDA deterministic operations. In particular, we use the `seed_everything` function from Pytorch-Lightning[10] to set the seed for pseudo-random number generators in `pytorch`, `numpy`, and `python.random`. In addition, it sets the following environment variables: `PL_GLOBAL_SEED` that is passed to spawned subprocesses (e.g. ddp_spawn backend) and `PL_SEED_WORKERS`.

Hyper-parameters. In our initial exploration, we used the default hyper-parameters suggested in the T5 paper (Raffel et al., 2020) and on the HuggingFace blog,[11] that is batch size 128 and constant learning rate equal to 10^{-3}. Given the size of MultiWoz, it roughly corresponded to 4k update steps. However, this budget proved to be insufficient. In particular, our own re-implementation of the cumulative state type of models was not in line with the results reported in the literature. More importantly, our *Diable* model was clearly undertrained as demonstrated by the fact that model selection on the validation set was consistently selecting the last checkpoint. Therefore, we scaled up the training budget by 10x to roughly 40k update steps. We rescaled the batch size to 32 and, consistently, the learning rate to 10^{-4}—a similar setup is used by Zhao et al. (2022). This new training budget corresponds to roughly 20 epochs. We did not notice any significant improvement by further increasing it. Finally, we experimented with both Adafactor and AdamW (Loshchilov and Hutter, 2019) and the former consistently outperformed the latter while also speeding up the training process.

D Complete Tables of Results

In this section, we report the complete set of results for our *Diable* system and *light*Cumulative (our own reproduction of a cumulative state model). We run each experiment with 5 different random seeds and report statistics across runs. Furthermore, we show the effect of label normalisation in rows

[10] https://github.com/Lightning-AI/lightning/blob/94e6d52b7e2f2a9ffc21f7e11e087808666fe710/src/lightning_lite/utilities/seed.py#L20

[11] https://discuss.huggingface.co/t/t5-finetuning-tips/684.

Contex	Dataset version	JGA
Cumulative state		
$D_{1:t}$	2.1	51.06
$D_{1:t}$	2.2	57.30
$D_{1:t}$	2.3	48.21
$D_{1:t}$	2.4	58.52
$D_{1:t}$	2.1 (fix labels)	51.94
$D_{1:t}$	2.2 (fix labels)	57.37
$D_{1:t}$	2.3 (fix labels)	49.77
$D_{1:t}$	2.4 (fix labels)	65.82
Diable		
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1	49.86
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2	56.42
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3	47.84
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4	58.00
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1 (fix labels)	50.92
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2 (fix labels)	56.74
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3 (fix labels)	49.70
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4 (fix labels)	65.01

Table 8: JGA on the evaluation sets of MultiWoz 2.1-2.4 for T5v1.1-large models trained on the MultiWoz 2.2 training set. Result statistics obtained across 5 random seeds. The evaluation also includes the raw metrics with no label normalisation.

identified by "fix label".

Table 8 shows the JGA on the evaluation sets of MultiWoz 2.1-2.4 for the T5v1.1-large models trained on the MultiWoz 2.2. Table 11 reports the JGA on the evaluation sets of MultiWoz 2.1-2.4 for T5v1.1-base models trained on the MultiWoz 2.1. Finally, Table 12 contains the JGA on the evaluation sets of MultiWoz 2.1-2.4 for T5v1.1-base models trained on the MultiWoz 2.2.

Slots	Number of dialogues				Number of turns			
	2.1	**2.2**	**2.3**	**2.4**	**2.1**	**2.2**	**2.3**	**2.4**
attraction-area	2397	2396	2431	2396	16232	16401	16521	16277
attraction-name	2270	2260	3348	2358	12557	12710	18348	13041
attraction-type	2502	2500	2542	2503	16900	17003	17203	16940
hotel-area	2416	2417	2478	2452	17213	17562	17457	17490
hotel-day	2599	2599	2620	2597	13927	13963	14141	13928
hotel-internet	1772	1772	1953	1786	12706	12920	13395	12814
hotel-name	3039	3542	3787	3154	16770	19789	21920	17431
hotel-parking	1819	1819	2001	1848	12903	13107	13617	13095
hotel-people	2605	2605	2631	2606	13889	13952	14123	13935
hotel-pricerange	2244	2246	2324	2251	16070	16337	16327	16146
hotel-stars	1984	1984	2035	1969	14220	14475	14435	14177
hotel-stay	2605	2605	2624	2605	13960	14024	14168	13993
hotel-type	2262	2263	2728	2242	16339	16733	17891	16344
restaurant-area	3414	3412	3461	3400	22915	23237	23119	22894
restaurant-day	2626	2626	2643	2626	14463	14485	14643	14492
restaurant-food	3605	3599	3647	3605	24545	24841	24822	24594
restaurant-name	3230	3850	4298	3358	16946	20249	22978	17690
restaurant-people	2635	2635	2655	2638	14543	14590	14747	14591
restaurant-pricerange	3331	3330	3381	3314	22498	22759	22793	22472
restaurant-time	2620	2616	2649	2621	14346	14410	14548	14374
taxi-arriveby	846	840	861	845	3122	3158	3190	3134
taxi-departure	1836	1826	1840	1832	7089	7087	7124	7077
taxi-destination	1837	1831	1841	1829	7091	7117	7149	7073
taxi-leaveat	1071	1051	1094	1063	3963	3929	4077	3934
train-arriveby	2111	2106	2136	2106	13097	13149	13220	13086
train-day	3793	3792	3828	3793	24656	24680	24854	24678
train-departure	3783	3774	3832	3768	24782	24792	25185	24704
train-destination	3806	3799	3831	3796	25254	25247	25458	25201
train-leaveat	2035	2023	2085	2038	12622	12838	12802	12753
train-people	2266	2266	2284	2225	10858	10932	11043	10700

Table 9: Frequency of slots across dialogues and turns for MultiWoz 2.1-2.4.

	Train				Validation				Test			
Version	**2.1**	**2.2**	**2.3**	**2.4**	**2.1**	**2.2**	**2.3**	**2.4**	**2.1**	**2.2**	**2.3**	**2.4**
Number of domains												
mean	1.81	1.81	1.82	1.81	1.97	1.96	1.97	1.95	1.93	1.93	1.94	1.92
std	0.69	0.68	0.69	0.69	0.62	0.62	0.63	0.61	0.63	0.62	0.63	0.62
min	1	1	1	1	1	1	1	1	1	1	1	1
25%	1	1	1	1	2	2	2	2	2	2	2	2
50%	2	2	2	2	2	2	2	2	2	2	2	2
75%	2	2	2	2	2	2	2	2	2	2	2	2
max	5	4	5	5	4	4	4	4	4	4	4	4
Number of turns												
mean	7.96	7.96	7.96	7.96	8.37	8.37	8.37	8.37	8.37	8.37	8.37	8.37
std	2.59	2.59	2.59	2.59	2.24	2.24	2.24	2.24	2.37	2.37	2.37	2.37
min	2	2	2	2	3	3	3	3	3	3	3	3
25%	6	6	6	6	7	7	7	7	7	7	7	7
50%	8	8	8	8	8	8	8	8	8	8	8	8
75%	10	10	10	10	10	10	10	10	10	10	10	10
max	23	23	23	23	18	18	18	18	19	19	19	19
Number of slots												
mean	7.49	7.59	7.95	7.49	7.97	8.10	8.44	8.23	8.08	8.16	8.45	8.08
std	3.51	3.59	3.62	3.51	3.13	3.23	3.23	3.26	3.20	3.27	3.27	3.20
min	1	1	1	1	1	1	1	1	1	1	1	1
25%	5	5	5	5	6	6	6	6	6	6	6	6
50%	7	7	8	7	8	8	8	8	8	8	8	8
75%	10	10	11	10	10	11	11	11	10	11	11	10
max	20	20	20	20	17	18	17	17	18	19	18	19

Table 10: Number of domains, turns, and slots per dialogue for MultiWoz 2.1-2.4.

Context	Dataset version	mean	std	min	25%	50%	75%	max
Cumulative state								
$D_{1:t}$	2.1	53.80	0.65	52.81	53.57	54.02	54.10	54.53
$D_{1:t}$	2.2	53.48	0.58	52.65	53.30	53.49	53.74	54.22
$D_{1:t}$	2.3	49.33	0.59	48.43	49.29	49.31	49.62	50.01
$D_{1:t}$	2.4	60.54	0.57	60.01	60.22	60.42	60.74	61.33
$D_{1:t}$	2.1 (fix labels)	53.91	0.63	52.96	53.74	54.07	54.11	54.67
$D_{1:t}$	2.2 (fix labels)	54.60	0.61	53.77	54.37	54.46	55.17	55.25
$D_{1:t}$	2.3 (fix labels)	51.16	0.40	50.66	50.84	51.29	51.38	51.63
$D_{1:t}$	2.4 (fix labels)	67.56	0.52	67.13	67.27	67.34	67.65	68.43
Diable								
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1	53.80	0.65	52.75	53.78	53.84	54.11	54.50
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2	53.86	0.41	53.31	53.65	53.87	54.11	54.38
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3	49.70	0.57	48.96	49.38	49.76	49.93	50.46
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4	62.80	0.99	61.06	63.13	63.14	63.19	63.48
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1 (fix labels)	53.97	0.66	52.90	53.92	54.07	54.33	54.65
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2 (fix labels)	54.58	0.67	53.51	54.48	54.63	55.02	55.25
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3 (fix labels)	51.65	0.46	51.04	51.33	51.76	52.06	52.08
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4 (fix labels)	70.46	1.18	68.49	70.60	70.66	70.90	71.64
$\mathcal{B}_{t-1}$	2.1	53.58	0.30	53.16	53.46	53.59	53.73	53.96
$\mathcal{B}_{t-1}$	2.2	53.03	0.23	52.78	52.81	53.08	53.20	53.30
$\mathcal{B}_{t-1}$	2.3	49.30	0.85	48.43	48.82	49.12	49.50	50.65
$\mathcal{B}_{t-1}$	2.4	61.48	0.83	60.85	61.01	61.26	61.33	62.93
$\mathcal{B}_{t-1}$	2.1 (fix labels)	53.91	0.70	53.31	53.49	53.72	53.99	55.07
$\mathcal{B}_{t-1}$	2.2 (fix labels)	54.41	0.93	53.73	53.91	54.02	54.35	56.02
$\mathcal{B}_{t-1}$	2.3 (fix labels)	51.48	0.70	51.07	51.13	51.17	51.29	52.73
$\mathcal{B}_{t-1}$	2.4 (fix labels)	70.03	0.95	68.79	69.25	69.76	70.55	71.42

Table 11: JGA on the test sets of MultiWoz 2.1-2.4 for T5v1.1-base models trained on the MultiWoz 2.1 training set. Result statistics obtained across 5 random seeds. The evaluation also includes the raw metrics with no label normalisation.

Context	Dataset version	mean	std	min	25%	50%	75%	max
Cumulative state								
$D_{1:t}$	2.1	49.47	0.22	49.12	49.39	49.55	49.63	49.66
$D_{1:t}$	2.2	56.72	0.49	56.17	56.38	56.61	57.18	57.28
$D_{1:t}$	2.3	47.67	0.59	47.10	47.19	47.48	48.14	48.44
$D_{1:t}$	2.4	56.89	0.51	56.16	56.65	57.04	57.08	57.50
$D_{1:t}$	2.1 (fix labels)	50.69	0.37	50.45	50.47	50.60	50.61	51.34
$D_{1:t}$	2.2 (fix labels)	57.01	0.45	56.43	56.81	56.89	57.42	57.51
$D_{1:t}$	2.3 (fix labels)	49.51	0.63	48.74	48.94	49.81	49.85	50.19
$D_{1:t}$	2.4 (fix labels)	63.11	0.83	61.98	62.89	63.13	63.27	64.30
$D_{1:t} + \mathcal{B}_{t-1}$	2.1	49.50	0.50	48.91	49.12	49.59	49.72	50.18
$D_{1:t} + \mathcal{B}_{t-1}$	2.2	56.43	0.73	55.41	56.20	56.28	56.96	57.28
$D_{1:t} + \mathcal{B}_{t-1}$	2.3	47.63	0.57	46.96	47.26	47.52	48.14	48.29
$D_{1:t} + \mathcal{B}_{t-1}$	2.4	57.54	0.55	56.76	57.15	57.84	57.95	57.99
$D_{1:t} + \mathcal{B}_{t-1}$	2.1 (fix labels)	50.71	0.55	50.04	50.47	50.57	50.98	51.51
$D_{1:t} + \mathcal{B}_{t-1}$	2.2 (fix labels)	56.89	0.57	56.19	56.65	56.67	57.33	57.60
$D_{1:t} + \mathcal{B}_{t-1}$	2.3 (fix labels)	49.45	0.54	48.81	49.08	49.50	49.67	50.20
$D_{1:t} + \mathcal{B}_{t-1}$	2.4 (fix labels)	63.27	0.68	62.11	63.27	63.48	63.74	63.75
$\mathcal{B}_{t-1}$	2.1	49.59	0.26	49.16	49.57	49.65	49.74	49.84
$\mathcal{B}_{t-1}$	2.2	56.38	0.46	55.68	56.20	56.43	56.69	56.88
$\mathcal{B}_{t-1}$	2.3	47.20	0.79	46.13	46.72	47.30	47.80	48.06
$\mathcal{B}_{t-1}$	2.4	57.35	1.21	55.22	57.54	57.89	57.99	58.13
$\mathcal{B}_{t-1}$	2.1 (fix labels)	50.90	0.11	50.77	50.83	50.91	50.95	51.06
$\mathcal{B}_{t-1}$	2.2 (fix labels)	56.50	0.47	55.93	56.13	56.57	56.81	57.08
$\mathcal{B}_{t-1}$	2.3 (fix labels)	49.08	0.90	47.90	48.48	49.23	49.65	50.15
$\mathcal{B}_{t-1}$	2.4 (fix labels)	63.52	0.96	62.22	62.89	63.73	64.24	64.53
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1	49.38	0.27	49.08	49.09	49.54	49.59	49.61
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2	56.82	0.92	55.48	56.25	57.23	57.49	57.64
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3	47.52	0.72	46.65	47.12	47.38	47.98	48.48
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4	57.71	0.86	56.73	57.01	57.68	58.49	58.64
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1 (fix labels)	50.58	0.34	50.11	50.34	50.76	50.77	50.90
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2 (fix labels)	57.12	0.93	55.70	56.69	57.45	57.81	57.95
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3 (fix labels)	49.58	0.97	48.03	49.27	50.03	50.09	50.46
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4 (fix labels)	63.15	1.56	61.58	61.96	62.70	64.23	65.26
Diable								
$D_{1:t}$	2.1	48.74	0.20	48.48	48.67	48.67	48.91	48.98
$D_{1:t}$	2.2	55.33	0.70	54.40	55.11	55.28	55.58	56.31
$D_{1:t}$	2.3	47.46	0.24	47.18	47.35	47.35	47.65	47.78
$D_{1:t}$	2.4	58.76	0.57	58.25	58.29	58.68	58.94	59.64
$D_{1:t}$	2.1 (fix labels)	50.00	0.14	49.81	49.88	50.08	50.08	50.14
$D_{1:t}$	2.2 (fix labels)	55.63	0.68	54.73	55.39	55.63	55.78	56.62
$D_{1:t}$	2.3 (fix labels)	49.31	0.26	49.09	49.13	49.21	49.40	49.72
$D_{1:t}$	2.4 (fix labels)	64.95	0.55	64.05	64.79	65.29	65.30	65.33
$D_{1:t} + \mathcal{B}_{t-1}$	2.1	48.86	0.30	48.49	48.68	48.82	49.04	49.27
$D_{1:t} + \mathcal{B}_{t-1}$	2.2	56.04	0.77	55.33	55.71	55.82	56.01	57.34
$D_{1:t} + \mathcal{B}_{t-1}$	2.3	48.53	0.66	47.82	48.17	48.40	48.71	49.57
$D_{1:t} + \mathcal{B}_{t-1}$	2.4	59.60	0.54	58.94	59.10	59.86	59.94	60.16
$D_{1:t} + \mathcal{B}_{t-1}$	2.1 (fix labels)	50.00	0.30	49.62	49.86	49.92	50.22	50.39
$D_{1:t} + \mathcal{B}_{t-1}$	2.2 (fix labels)	56.28	0.84	55.44	55.87	56.19	56.24	57.66
$D_{1:t} + \mathcal{B}_{t-1}$	2.3 (fix labels)	50.57	0.73	49.74	50.23	50.38	50.79	51.70
$D_{1:t} + \mathcal{B}_{t-1}$	2.4 (fix labels)	66.51	0.87	65.18	66.35	66.56	66.93	67.53
$\mathcal{B}_{t-1}$	2.1	49.03	0.24	48.74	48.93	48.96	49.17	49.35
$\mathcal{B}_{t-1}$	2.2	55.99	0.66	55.40	55.67	55.81	55.94	57.12
$\mathcal{B}_{t-1}$	2.3	47.81	0.34	47.25	47.82	47.90	47.94	48.14
$\mathcal{B}_{t-1}$	2.4	59.13	0.86	58.19	58.42	59.10	59.73	60.23
$\mathcal{B}_{t-1}$	2.1 (fix labels)	50.58	0.52	50.15	50.23	50.45	50.62	51.45
$\mathcal{B}_{t-1}$	2.2 (fix labels)	56.30	0.67	55.63	55.98	56.21	56.24	57.42
$\mathcal{B}_{t-1}$	2.3 (fix labels)	49.85	0.40	49.20	49.88	49.88	49.96	50.31
$\mathcal{B}_{t-1}$	2.4 (fix labels)	66.13	0.97	65.04	65.45	66.03	66.63	67.49
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1	49.27	0.42	48.82	48.87	49.31	49.67	49.69
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2	56.21	0.56	55.67	55.79	56.00	56.62	56.96
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3	48.01	0.52	47.27	47.83	47.98	48.36	48.62
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4	58.86	0.31	58.53	58.68	58.76	58.98	59.32
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.1 (fix labels)	50.40	0.38	49.93	50.07	50.50	50.72	50.77
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.2 (fix labels)	56.48	0.57	55.90	56.02	56.36	56.88	57.26
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.3 (fix labels)	50.00	0.63	49.04	49.86	50.05	50.31	50.73
$D_{t-4:t} + \mathcal{B}_{t-1}$	2.4 (fix labels)	65.35	0.50	64.68	65.19	65.37	65.48	66.05
Gold previous state								
$\mathcal{B}_{t-1}$	2.1	68.44	0.14	68.20	68.41	68.49	68.50	68.58
$\mathcal{B}_{t-1}$	2.2	82.38	0.27	81.96	82.28	82.43	82.53	82.68
$\mathcal{B}_{t-1}$	2.3	58.49	0.20	58.25	58.40	58.46	58.56	58.79
$\mathcal{B}_{t-1}$	2.4	55.38	0.25	55.20	55.20	55.21	55.60	55.71
$\mathcal{B}_{t-1}$	2.1 (fix labels)	70.00	0.16	69.74	69.95	70.05	70.10	70.14
$\mathcal{B}_{t-1}$	2.2 (fix labels)	82.50	0.28	82.07	82.45	82.54	82.65	82.81
$\mathcal{B}_{t-1}$	2.3 (fix labels)	61.06	0.22	60.78	60.95	61.04	61.19	61.34
$\mathcal{B}_{t-1}$	2.4 (fix labels)	61.46	0.32	61.18	61.22	61.29	61.81	61.81

Table 12: JGA on the test sets of MultiWoz 2.1-2.4 for T5v1.1-base models trained on the MultiWoz 2.2 training set. Result statistics obtained across 5 random seeds. The evaluation also includes the raw metrics with no label normalisation.