MoNET: Tackle State Momentum via Noise-Enhanced Training for Dialogue State Tracking

Haoning Zhang\textsuperscript{1,3}, Junwei Bao\textsuperscript{2*}, Haipeng Sun\textsuperscript{2}, Youzheng Wu\textsuperscript{2}, Wenye Li\textsuperscript{1,5}, Shuguang Cui\textsuperscript{3,1,6}, Xiaodong He\textsuperscript{2}

\textsuperscript{1}FNii, CUHK-Shenzhen \textsuperscript{2}JD AI Research \textsuperscript{3}SSE, CUHK-Shenzhen \textsuperscript{4}SDS, CUHK-Shenzhen \textsuperscript{5}SRIBD \textsuperscript{6}Pengcheng Lab

haoningzhang@link.cuhk.edu.cn, \{wyli, shuguangcui\}@cuhk.edu.cn, \{baojunwei, sunhaipeng6, wuyouzheng1, xiaodong.he\}@jd.com

Abstract

Dialogue state tracking (DST) aims to convert the dialogue history into dialogue states which consist of slot-value pairs. As condensed structural information memorizes all dialogue history, the dialogue state in the previous turn is typically adopted as the input for predicting the current state by DST models. However, these models tend to keep the predicted slot values unchanged, which is defined as state momentum in this paper. Specifically, the models struggle to update slot values that need to be changed and correct wrongly predicted slot values in the previous turn. To this end, we propose MoNET to tackle state momentum via noise-enhanced training. First, the previous state of each turn in the training data is noised via replacing some of its slot values. Then, the noised previous state is used as the input to learn to predict the current state, improving the model’s ability to update and correct slot values. Furthermore, a contrastive context matching framework is designed to narrow the representation distance between a state and its corresponding noised variant, which reduces the impact of noised state and makes the model better understand the dialogue history. Experimental results on MultiWOZ datasets show that MoNET outperforms previous DST methods. Ablations and analysis verify the effectiveness of MoNET in alleviating state momentum issues and improving the anti-noise ability\textsuperscript{1}.

1 Introduction

Dialogue state tracking (DST) is a core component in modular task-oriented dialogue systems (Hosseini-Asl et al., 2020; Yang et al., 2021; Sun et al., 2022, 2023). It extracts users’ intents from the dialogue history and converts them into structural dialogue states, i.e., sets of slot-value pairs. An accurate dialogue state is crucial for generating correct dialogue action and suitable natural language responses, which are the main tasks of dialogue management and natural language generation components (Williams and Young, 2007; Thomson and Young, 2010; Young et al., 2010). Earlier DST approaches predict the state directly from the dialogue history (natural language utterances) (Mrkšić et al., 2017; Xu and Hu, 2018; Wu et al., 2019; Chen et al., 2020a). Since the dialogue state is condensed structural information memorizing all dialogue history, recent methods incorporate the previously predicted state as the input besides the dialogue history (Ouyang et al., 2020; Kim et al., 2020; Ye et al., 2021).
Conventional DST models taking the previous state as the input usually show the characteristic that the previously predicted slot values tend to be kept unchanged when predicting the current state, defined as state momentum in this paper. The state momentum makes DST models struggle to modify the previous prediction, which affects the performance when the values of some slots need to be updated as the user’s intent changes, and there exist wrongly predicted slot values that need to be corrected. Figure 1 gives an example of a dialogue involving three turns with two types of state momentum issues. The state hotel-book day-Saturday is predicted in Turn 1 and keeps unchanged in the next two turns, while the user’s request is updated into Sunday in Turn 2. Consequently, the predicted state becomes wrong in the following two turns. The dotted arrow represents the ideal prediction cases: the value is updated with the ground truth changes and is corrected when becoming a wrong input. The solid arrow represents the state momentum issues, where the state is kept unchanged, leading to two consecutive wrong predictions. One possible reason for the state momentum issue is that in the training data, most slot values in the previous turn are the same as those in the current turn, which limits the ability of conventional DST models to modify slot values during inference. To address this limitation, an intuitive idea is to augment training instances with a higher ratio of slots whose previous values differ from those in the current turn. By incorporating such examples, the DST model can learn to deal with more cases where modifying previous predictions is required. Besides, if the DST model can treat wrong and correct dialogue states similarly in representations, then the former will typically help make further predictions. In other words, by treating incorrect dialogue states as valuable information, the DST model can potentially identify and correct erroneous slot values.

In this paper, we propose MoNET to tackle the state momentum issue via a noise-enhanced training strategy. The core idea is to manually add noise into the previous state to simulate scenarios with wrong state input. First, the previous state of each turn in the training data is noised via replacing some of its slot values. Specifically, for each active slot (with a non-none value), we replace its value with a certain probability. Then, the noised previous state, concatenated with the dialogue history, is used as the input to learn to predict the current state, improving the model’s ability to update and correct slot values. Furthermore, a contrastive context matching framework is designed to narrow the representation distance between a state and its corresponding noised variant, which reduces the impact of the noised state and makes the DST model better understand the dialogue history. Such approaches make the model less sensitive to the noise, and enhance its ability to modify the slot values of previous states in current predictions. Experiments on the multi-domain dialogue datasets MultiWOZ 2.0, 2.1, and 2.4 show that our MoNET outperforms previous DST models. Ablation studies and analysis further verify the effectiveness of the proposed noised DST training and the contrastive context matching framework in alleviating state momentum and improving the model’s anti-noise ability.

The contributions are summarized as follows:
1. We define the state momentum issue in DST, where models tend to keep the predicted slot values unchanged, namely, struggling to update and correct them from the previous turn.
2. We propose MoNET to tackle the state momentum issue via noised DST training and the contrastive context matching framework.
3. We conduct comprehensive experiments on three datasets, MultiWOZ 2.0, 2.1, and 2.4. The results demonstrate that MoNET outperforms previous DST methods, showcasing its effectiveness in alleviating the state momentum issue.

2 Related Work

2.1 Dialogue State Tracking

Traditional DST approaches focus on single-domain dialogue state tracking (Williams and Young, 2007; Thomson and Young, 2010; Lee and Kim, 2016). Recent researches pay more attention to multi-domain DST using distributed representation learning (Wen et al., 2017; Mrkšič et al., 2017). Previous works implement Seq2seq frameworks to encode the dialogue history, then predict the dialogue state from scratch at every turn (Rastogi et al., 2017; Ren et al., 2018; Lee et al., 2019; Wu et al., 2019; Chen et al., 2020a). Utilizing dialogue history is limited for larger turns, since the state of each turn is accumulated from all previous turns, while it’s hard to retrieve state information from a long history.
Current works mainly incorporate the previous state as the model input, which is regarded as an explicit fixed-sized memory (Ouyang et al., 2020; Ye et al., 2022a; Wang et al., 2022). Kim et al. (2020) propose a state operation sub-task, where the model is trained to first predict the operation of each slot-value pair, such as UPDATE, CARRYOVER, etc., then only the value of a minimal subset of slots will be newly modified (Zeng and Nie, 2020; Zhu et al., 2020). These methods enhance model prediction efficiency and the ability to update slot-value pairs. Tian et al. (2021) deal with the error propagation problem that mistakes are prone to be carried over to the next turn, and design a two-pass generation process, where a temporary state is first predicted then used to predict the final state, enhancing the ability to correct wrong predictions. In this paper, we use “state momentum” to define the issue where the wrong dialogue state is predicted due to that the previous prediction keeps unchanged, either it should be updated or corrected. To the best of our knowledge, this is the first time to systematically tackle the issue caused by continuous unchanged predictions in the multi-turn DST task.

2.2 Contrastive Learning

Contrastive learning aims to generate high-quality representations by constructing pairs of similar examples to learn semantic similarity (Mnih and Teh, 2012; Baltescu and Blunsom, 2015; Peters et al., 2018). The goal is to help the model semantically group similar instances together and separate dissimilar instances. During training, the neighbors with similar semantic representations (positive pairs) will be gathered, while the non-neighbors (negative pairs) will be pushed apart, enabling the learning of more meaningful representations. In the NLP area, semantic representations can be learned through self-supervised methods, such as center word prediction in Word2Vec, next sentence prediction in BERT, sentence permutation in BART, etc (Mikolov et al., 2013; Devlin et al., 2019; Lewis et al., 2020). Recent approaches build augmented data samples through token shuffling, word deletion, dropout, and other operations (Cai et al., 2020; Klein and Nabi, 2020; Yan et al., 2021; Wang et al., 2021; Gao et al., 2021; Zhang et al., 2022). In this paper, we construct augmented samples based on the noised and original dialogue state. Given context inputs with the same dialogue history and different states, the model is trained to gather them into similar objects, aiming to learn better representations, reduce the impact of noise, and better understand the dialogue history.

3 Methodology

3.1 Problem Formulation

In this paper, we focus on building a dialogue state tracking (DST) model which accurately predicts the dialogue state based on the dialogue history and the previous state during multi-turn dialogue interactions. A dialogue state consists of domain-slot-value tuples, typically corresponding to the dialogue topic, the user’s goal, and the user’s intent. Following previous studies, in the rest of this paper, we omit “domain” and use “slot” to refer to a “domain-slot” pair. All slot-value pairs are from a pre-defined ontology.

Formally, let’s define $D_t = [U_t, R_t]$ as a pair of system utterance $U_t$ and user response $R_t$ in the $t$-th turn of a multi-turn dialogue, and $B_t$ as the corresponding dialogue state. Each state $B_t$ contains a set of slot-value pairs, i.e., $B_t = \{(S_j, V^j) | j \in [1:J]\}$, where $J$ is the total number of slots, and $V^j \in V_j$ is one of the values in $V_j$ for the $j$-th slot $S_j$ in the ontology. Given the dialogue history $\{D_1, \ldots, D_t\}$ and previous state $B_{t-1}$, the goal of the DST task is to predict the current dialogue state $B_t$.

3.2 MoNET

As introduced in Section 1, solving the state momentum issue is crucial to the DST task. Therefore, in this paper, we propose MoNET to tackle the state momentum issue via a noise-enhanced training strategy to enhance the model’s ability to update and correct slot values. The architecture of MoNET is shown in Figure 2(a), which consists of context BERT encoders, slot and value BERT encoders, the slot-context attention module, the slot-value matching module, and the contrastive context matching framework. Each of them will be elaborated on in this section.

3.2.1 Base Architecture

We first introduce the base architecture of our MoNET, similar to the backbone model in (Ye et al., 2022a). A model trained only with the base architecture of MoNET is noted as “Baseline”, and
evaluated in Section 5 to compare the difference in performance with the whole MoNET model.

**Context Encoder.** A BERT encoder encodes the context input, which is the concatenation of the dialogue history and the state in the previous turn:

\[ X_t = f(M_t, B_{t-1}, D_t) = [CLS] \oplus M_t \oplus B_{t-1} \oplus [SEP] \oplus D_t \oplus [SEP], \]

where \( M_t = D_1 \oplus \ldots \oplus D_{t-1} \) contains previous utterances, \( B_{t-1} \) is the state containing the active slots in the previous turn, \([CLS]\) and \([SEP]\) are special tokens of the BERT encoder. Then the representations of the context input are derived:

\[ H_t = BERT(X_t) \in \mathbb{R}^{[X_t] \times d}, \]

where \([X_t]\) is the total number of tokens in \( X_t \), and \( d \) is the encoded hidden size.

**Slot and Value Encoders.** The BERT encoders with fixed parameters are used to derive the slot and value representations:

\[ h_{S_j} = BERT_{fixed}(S_j)[CLS], \]

\[ h_{V_j} = BERT_{fixed}(V_j)[CLS], \]

where states \( h_{S_j}, h_{V_j} \in \mathbb{R}^{1 \times d} \) are the \([CLS]\) representations of the slot and value.

**Slot-Context Attention.** For each slot \( S_j \), its slot-context-specific feature is extracted by the multi-head attention mechanism (Vaswani et al., 2017):

\[ r^s_j = LN(MultiHead(h_{S_j}, H_t, H_t)) \in \mathbb{R}^{1 \times d}, \]

where \( LN \) is the normalization layer.

**Slot-Value Matching.** The probability of predicting the value \( V_j \) of the slot \( S_j \) is derived by calculating the \( L2\)-distance between the value representation \( h_{V_j} \) and the slot-context representation \( r^s_j \), which is denoted as:

\[ P_{\theta}(V_j|X_t, S_j) = \frac{\exp(-||r^s_j - h_{V_j}||^2_2)}{\sum_{k \in [1:|V_j|]} \exp(-||r^s_j - h_{V_k}||^2_2)}, \]

where \( \theta \) are trainable parameters of the model.

**Training and Inference.** During training, the ground dialogue state is used to form the context input \( X_t \) (teacher-forcing). For the \( t\)-th turn, the loss is the sum of the negative log-likelihood among all \( J \) slots as follows:

\[ L_{	ext{dori}} = \sum_{j=1}^{J} -\log(P_{\theta}(V_j^*|X_t, S_j)), \]
where $V_j^t$ is the ground truth value of the slot $S_j$ at turn $t$. During inference, the previously predicted state is used to form the context input $X_t$, and the value of the slot $S_j$ is predicted by selecting the one with the smallest distance, corresponding to the largest probability:

$$V_j^t = \arg\max_{i \in [1:|V_j|]} P_{\theta}(V_j^t | X_t, S_j).$$

### 3.2.2 Noised Data Construction

As described previously, an intuitive idea to tackle the state momentum issue is to increase the number of training instances where the slot-value pairs in the previous turn are different from those in the current turn. Based on this point, we attempt to utilize noised data to train the DST model.

Generally, for each active slot (with a non-none value) in the previous dialogue state, we involve noise by replacing its original value with another value with a probability $p$ (used as the noise threshold), e.g., as the example shown in Figure 2(b). Formally, at each training step, given a batch of training instances, a noised context input $X_t^+$ is constructed for each instance based on its original context input $X_t = f(M_t, B_{t-1}, D_t)$ as follows:

$$X_t^+ = f(M_t, B_{t-1}, D_t),$$
$$B_{t-1}^- = \{(S_j, V_j^t) | j \in [1:J]\}.$$

For each active slot $S_j$ in $B_{t-1}^- = \{(S_j, V_j^t)\}$, a real number $a \in [0, 1]$ is sampled to determine whether the original $V_j^t$ is replaced with a randomly selected value $V_j^k \in V_j \setminus \{V_j^t\}$ from the ontology or kept unchanged:

$$V_j^k = \begin{cases} V_j^k, & \text{if } a < p \\ V_j^t, & \text{if } a \geq p. \end{cases}$$

### 3.2.3 Noised State Tracking

Similar to $X_t$, the noised context instance $X_t^+$ is also used as the model input to predict the state $B_t$ as the training target, aiming to improve the model’s ability to dynamically modify the previous slot values in current predictions. Specifically, the representation $H_t^+$ of $X_t^+$ is first derived by the BERT context encoder mentioned in Section 3.2.1:

$$H_t^+ = \text{BERT}(X_t^+),$$

Then, similar to the previous process, for each slot $S_j$, $X_t^+$ is used to predict its value based on the distribution $P_{\theta}(V_j^t | X_t^+, S_j)$. Eventually, the loss for the noised state tracking can be denoted as:

$$L_{d_n} = \sum_{j=1}^{J} -\log(P_{\theta}(V_j^t | X_t^+, S_j)).$$

### 3.2.4 Contrastive Context Matching

Inspired by contrastive learning approaches which group similar samples closer and diverse samples far from each other, a contrastive context matching framework is designed to narrow the representation distance between $X_t$ and its noised variant $X_t^+$, aiming to reduce the impact of the noised state $B_{t-1}^-$ and help the model better understand the dialogue history. Specifically, in a batch of $N$ instances with the original context input $X_t = \{X_t^n\}_{n=1}^N$, we construct $N$ corresponding noised instances with the context input $X_t^+ = \{X_t^{n+}\}_{n=1}^N$. To clearly describe the context inputs, in this section, we temporarily involve $n$ into $X_t$ & $H_t$ as $X_t^n$ & $H_t^n$ to indicate the in-batch index.

For each context input $X_t^n$, its noised sample $X_t^{n+}$ is regarded as its positive pair, and the rest $(2N-2)$ instances in the same batch with different dialogue histories are considered negative pairs. Then the model is trained to narrow the distance of the positive pair and enlarge the distance of negative pairs in the representation space with the following training objective:

$$L_c = -\log\left(\frac{\exp(sim(H_t^n, H_t^{n+})/\tau)}{\sum_{i=1}^{N} \exp(sim(H_t^n, H_t^{i})/\tau)}\right),$$

where $H_t^{n+}$ and $H_t^n$ are the [CLS] representations of $H_t^n$ and $H_t^{n+}$, $\tau$ is the temperature parameter, and $sim(\cdot)$ indicates the cosine similarity function (Chen et al., 2020c).

### 3.2.5 Optimization

The total training loss for each instance is the sum of losses from the slot-value matching for DST and the contrastive context matching for representation learning, where the former is the average of the losses using the original or the noised context input mentioned in Section 3.2.1 and 3.2.3:

$$L_{tot} = (L_{d_ori} + L_{d_n})/2 + L_c.$$
2018) is a standard human-human conversational dialogue corpus with seven domains. MultiWOZ 2.1 (Eric et al., 2020) has the same dialogues as the 2.0 version, where some incorrect state labels are re-annotated. Both of them are widely used in previous DST approaches. MultiWOZ 2.4 (Ye et al., 2022b) is the latest refined version correcting all the incorrect state labels in validation and test sets. All three datasets contain the same number of dialogues, which are 8438/1000/1000 in train/validation/test sets. For the three datasets, we follow the previous work (Wu et al., 2019) to use five domains (attraction, hotel, restaurant, taxi, train) with 30 domain-slot pairs in experiments, since the dialogues in the remaining domains are not in the validation and test sets.

### 4.2 Evaluation Metrics

We use joint and slot goal accuracy as the evaluation metrics. Joint goal accuracy is the ratio of dialogue turns where the values of all slots are correctly predicted. Slot goal accuracy is the ratio of domain-slot pairs whose values are correctly predicted. Both of them include correctly predicting those inactive slots with the value \textit{none}.

### 4.3 Existing Methods

We compare the performance of our MoNET with several existing methods, i.e., TRADE, SUMBT, PIN, SOM-DST, CSFN-DST, DST-Picklist, SAVN, SST, SimpleTOD, TripPy, STAR, SDP-DST, DS-Graph, DSGFNet, PPTOD shown in Table 1, and our base architecture mentioned in Section 3.2.1, denoted as Baseline.

### 4.4 Training Details

The BERT-base-uncased model is used as the context, slot and value encoders, with 12 attention layers and a hidden size of 768. During training, only the parameters of the context BERT encoder are updated, while the parameters of the slot and value BERT encoders are not. The batch size is set to 8. The AdamW optimizer is applied to optimize the model with the learning rate 4e-5 and 1e-4 for the context encoders and the remaining modules, respectively (Loshchilov and Hutter, 2019). The temperature parameter $\tau$ is set to 0.1. The noise threshold $p$ defined in Section 3.2.2 is set to 0.3, and its impact on model performance is discussed in Section 5. All models are trained on a P40 GPU device for 6-8 hours.

### 5 Results and Analysis

#### 5.1 Main Results

Table 1 shows performances of MoNET and baselines on MultiWOZ 2.0, 2.1 and 2.4. Among them, TripPy and its modified versions employ a ground truth label map of synonyms replacement as extra supervision, which increases their accuracy scores and differs from other methods of testing with common labels. As can be observed, MoNET achieves the joint goal accuracy scores of 55.48 and 76.02 on MultiWOZ 2.1 and 2.4, respectively.

<table>
<thead>
<tr>
<th>Baseline</th>
<th>Pre-trained Model</th>
<th>MultiWOZ 2.0</th>
<th>MultiWOZ 2.1</th>
<th>MultiWOZ 2.4</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADE (Wu et al., 2019)</td>
<td>BERT-base</td>
<td>48.62</td>
<td>96.92</td>
<td>45.60</td>
</tr>
<tr>
<td>SUMBT (Lee et al., 2019)</td>
<td>BERT-base</td>
<td>42.40</td>
<td>49.01</td>
<td>96.76</td>
</tr>
<tr>
<td>PIN (Chen et al., 2020a)</td>
<td>BERT-base</td>
<td>52.44</td>
<td>97.28</td>
<td>48.40</td>
</tr>
<tr>
<td>SOM-DST (Kim et al., 2020)</td>
<td>BERT-base</td>
<td>51.72</td>
<td>53.01</td>
<td>-</td>
</tr>
<tr>
<td>CSFN-DST (Zhu et al., 2020)</td>
<td>BERT-base</td>
<td>52.23</td>
<td>53.19</td>
<td>-</td>
</tr>
<tr>
<td>DST-Picklist (Zhang et al., 2020)</td>
<td>BERT-base</td>
<td>54.39</td>
<td>53.30</td>
<td>97.40</td>
</tr>
<tr>
<td>SAVN (Wang et al., 2020)</td>
<td>BERT-base</td>
<td>54.52</td>
<td>97.42</td>
<td>54.86</td>
</tr>
<tr>
<td>SST (Chen et al., 2020b)</td>
<td>BERT-base</td>
<td>51.17</td>
<td>55.23</td>
<td>-</td>
</tr>
<tr>
<td>SimpleTOD (Hosseini-Asl et al., 2020)</td>
<td>DistilGPT2</td>
<td>-</td>
<td>-</td>
<td>55.26</td>
</tr>
<tr>
<td>Seq2SeqDU (Feng et al., 2021)</td>
<td>BERT-base</td>
<td>54.53</td>
<td>56.10</td>
<td>97.59</td>
</tr>
<tr>
<td>STAR (Ye et al., 2021)</td>
<td>BERT-base</td>
<td>54.86</td>
<td>97.47</td>
<td>-</td>
</tr>
<tr>
<td>SDP-DST (Lee et al., 2021)</td>
<td>T5-base</td>
<td>-</td>
<td>-</td>
<td>56.66</td>
</tr>
<tr>
<td>DS-Graph (Lin et al., 2021)</td>
<td>BERT-base</td>
<td>54.86</td>
<td>97.47</td>
<td>-</td>
</tr>
<tr>
<td>DSGFNet (Feng et al., 2022)</td>
<td>BERT-base</td>
<td>54.86</td>
<td>97.47</td>
<td>-</td>
</tr>
<tr>
<td>PPTOD (Su et al., 2022)</td>
<td>T5-large</td>
<td>53.89</td>
<td>-</td>
<td>57.45</td>
</tr>
</tbody>
</table>

Use Modified Label

| TripPy (Hock et al., 2020) | BERT-base | - | - | 55.29 | - | - |
| TripPy + SCoRe (Yu et al., 2021) | BERT-base | - | - | 60.48 | - | - |
| TripPy + CoCoAug (Li et al., 2021) | BERT-base | - | - | 60.53 | - | - |
| TripPy + SaCLog (Dai et al., 2021) | BERT-base | - | - | 60.61 | - | - |

Table 1: Joint and slot goal accuracy of our MoNET and several previous methods on three MultiWOZ test sets.
Table 2: Joint goal accuracy on the MultiWOZ 2.4 test set of MoNET and four ablated modifications.

<table>
<thead>
<tr>
<th>Model</th>
<th>NoisedCM</th>
<th>NoisedST</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>Baseline w/o state</td>
<td>×</td>
<td>×</td>
<td>64.94</td>
</tr>
<tr>
<td>Baseline</td>
<td>×</td>
<td>×</td>
<td>73.81</td>
</tr>
<tr>
<td>MoNET-ST</td>
<td>×</td>
<td>✓</td>
<td>75.54</td>
</tr>
<tr>
<td>MoNET-CM</td>
<td>✓</td>
<td>×</td>
<td>75.76</td>
</tr>
<tr>
<td>MoNET</td>
<td>✓</td>
<td>✓</td>
<td>76.02</td>
</tr>
</tbody>
</table>

55.48%, 57.71%, 76.02% in three datasets, which are impressive results compared with previous methods, and has improvements of 1.10%, 1.89%, and 2.21% on the Baseline model, indicating that our proposed noise-enhanced training helps the model make better predictions.

Besides the general joint and slot goal accuracy, we also calculate the slot-level proportion of state momentum errors over all wrong predictions. We train the Baseline model and make predictions on the MultiWOZ 2.4 test set. For each dialogue, starting from the second turn, we add up each wrong predicted slot-value pair which also exists in the previous turn. Finally, there are 844 such wrong slot-value pairs, and the number of all the wrong predicted pairs is 2603, hence the proportion is (844/2603)*100%≈32.4%, and our MoNET model modifies 47.0% of them (397 in 844 are correctly predicted). Moreover, in MultiWOZ 2.4 training set annotations, for each dialogue turn (also except the first turn of each dialogue), around 78.1% slot-value pairs exist in the previous turn, since the slot-value pairs will be accumulated as the dialogue progresses. The results further indicate the issue caused by those unchanged slot-value pairs during multi-turn interactions, and the effectiveness of our method in enhancing the model’s ability to modify previous predictions.

5.2 Ablation Study

To explore the individual contribution of each part of our model, we compare the whole MoNET with several ablated versions. First, we remove the previous dialogue state from the context input of the Baseline model, where the modified context input is \( X_t = [CLS] \oplus M_t \oplus [SEP] \oplus D_t \oplus [SEP] \), denoted as Baseline w/o state; besides, the two noise-enhanced methods are removed from the MoNET respectively, denoted as MoNET-CM (context matching only) and MoNET-ST (noised state tracking only).

Table 2 shows the joint goal accuracy performances of the full MoNET model and its four modifications on the MultiWOZ 2.4 test set. As can be observed, Baseline w/o state gets the lowest accuracy, demonstrating that explicitly using the previous dialogue state as part of the model input is beneficial to make predictions, even though there may exist wrong slot-value pairs. Besides, both MoNET-CM and MoNET-ST outperform the Baseline model, demonstrating functionalities of the noised state tracking in modifying slot-value pairs in further turns, and the context matching framework in learning improved semantic representations. Moreover, MoNET derives the best performance, demonstrating the effectiveness of integrating the two parts into a unified noise-enhance training strategy.

5.3 Turn-Level Evaluation

Figure 3 shows the turn-level joint goal accuracy of MoNET and Baseline models, as well as the percentage difference in accuracy (the difference between the two models’ accuracy divided by the accuracy of Baseline) on the MultiWOZ 2.4 test set. Generally, the state momentum issue becomes more apparent in dialogues with larger turns, since they always contain more active slot-value pairs, and any one of the wrong pairs kept unchanged will affect the further prediction accuracy. With the increase of turns, the accuracy of Baseline harshly degrades, while MoNET gets a relatively smaller decline, resulting in a gradually increasing and evident percentage difference in accuracy. This demonstrates the superiority of MoNET in alleviating the accuracy decrease caused by the state momentum issue, especially in those dialogues with larger than 6-7 turns.
### 5.4 Noise Threshold Selection for Training

To explore the impact of different probabilities of adding noise into the context input for training, we vary the noise threshold \( p \) from 0 to 0.5 to train our MoNET. The results on the MultiWOZ 2.4 validation set are shown in Figure 4, where MoNET achieves the best performance when the noise threshold \( p \) is set to 0.3. Intuitively, a small \( p \) makes the noised context input contain fewer noised slot-value pairs (hard to learn meaningful semantics from the noised data); conversely, a large \( p \) makes the noised context input far from the original context input in the representation space (hard to group them closer). Both two cases make the model hard to learn effective features from the noised context input, leading to lower prediction accuracy. Hence, the empirical probability of adding noise is important to derive the best performance of the DST model.

### 5.5 Anti-noise Probing with Noise Testing

In this section, we conduct noise testing to explore the impact of anti-noise ability on DST models. We first evaluate DST performances of MoNET and Baseline by introducing different ratios of noise (with \( p \) from 0 to 1) into the oracle previous dialogue state as the model input. Figure 5 shows the performances of MoNET and Baseline on MultiWOZ 2.4. Both of them get high accuracy when the noise ratio is 0, as we use the oracle previous dialogue state as the model input; with the increase of the noise ratio, the joint goal accuracy of Baseline gets a sharp decline, while MoNET degrades much more smoothly. Furthermore, for each dialogue turn, we also show the \( L_2 \)-distance between the original and noised context representations, i.e., the mean pooling of all token representations \( H_t \) and \( H_t^+ \). As can be observed, along with the increase of noise ratio, the distance between the two representations of MoNET is much lower than that of Baseline. These results indicate that MoNET achieves a higher anti-noise ability by generating relatively similar representations for the original and noised contexts, which helps the DST model maintain an acceptable performance even with a high ratio of noise in its input.

### 5.6 Case Study and Attention Visualization

Table 3 gives two prediction examples using MoNET and Baseline on the MultiWOZ 2.4 test set, corresponding to the two types of state momentum cases. In the first one, they correctly predict the slot-value pair “train-day-Sunday” and “taxi-destination-Gonville and Caius College”, respectively. In the second one, they make a wrong prediction “taxi-destination-Gonville and Caius College”. While
Baseline keeps it unchanged in the next turn, MoNET corrects it, resulting in a joint goal accuracy of 100% for the second turn. Besides, we further explore these two examples by calculating and visualizing the overall attention scores, which are shown in Figure 6. For each slot, its overall attention score over each token is the weighted sum of the self-attended scores by all tokens in $X_t$. The weights come from the slot-context attention, and the self-attended scores are the average of attention scores over multiple layers in BERT. As can be observed, Baseline pays more attention to the values in the previously predicted state, and fails to solve the state momentum issues; MoNET pays relatively higher attention to the correct tokens (“Monday” in the first case and “autumn house” in the second case), and consequently, successfully updates Sunday into Monday and corrects Gonville and Caius College into Autumn House. These examples and attention visualizations indicate the effectiveness of our MoNET in alleviating the two types of state momentum issues.

5.7 Extension on Generation-based Models

In addition to the original classification-based MoNET model, we also evaluate our approach using a simple generation framework using T5-base as the backbone pre-trained model (Raffel et al., 2020). The ontology is built from the database and training set annotations, which is only used for noise value construction. The model framework is similar to the BERT-based MoNET in Figure 2(a), where the BERT encoders and slot-value matching modules are replaced with T5 encoders and decoders. The T5 encoders encode the dialogue context inputs, slots, and values. After deriving the slot-context attentive representations, the T5 decoders generate each slot-value pair. Table 4 shows the joint goal accuracy performance of the T5-based MoNET on the MultiWOZ 2.0 test set, compared with other end-to-end/generation-based models using the same T5-base pre-trained model. As can be observed, our modified MoNET outperforms the T5-base backbone and others with the same T5-base model, indicating its effectiveness and adaptability for the implementation of generation-based methods.

<table>
<thead>
<tr>
<th>Model</th>
<th>Accuracy</th>
</tr>
</thead>
<tbody>
<tr>
<td>MinTL (Lin et al., 2020)</td>
<td>52.07</td>
</tr>
<tr>
<td>MTTOD (Lee, 2021)</td>
<td>53.56</td>
</tr>
<tr>
<td>PPTOD (Su et al., 2022)</td>
<td>53.37</td>
</tr>
<tr>
<td>T5-base</td>
<td>53.26</td>
</tr>
<tr>
<td>MoNET (T5-base)</td>
<td>54.67 (↑ 1.41)</td>
</tr>
</tbody>
</table>

Table 4: Joint goal accuracy on MultiWOZ 2.0 test set of baselines using the same T5-base pre-trained model.

6 Conclusion

In this paper, we define and systematically analyze the state momentum issues in the DST task, and propose MoNET, a training strategy equipped with noised DST training and the contrastive context matching framework. Extensive experiments on MultiWOZ 2.0, 2.1, and 2.4 datasets verify its effectiveness compared with existing DST methods. Supplementary studies and analysis demonstrate that MoNET has a stronger anti-noise ability which helps alleviate the state momentum issues.

Limitations

Our proposed MoNET is a classification-based method requiring the pre-defined ontology containing all slot-value pairs. Moreover, during prediction, for each slot, its distance with all possible values is calculated, i.e., the prediction has to process 30 times, which is the number of slots in the MultiWOZ dataset. Compared
with the generation methods that only process once and do not need ontology, our method is short in training efficiency and scalability. However, most task-oriented dialogue datasets contain their knowledge base containing slot value information, so it’s acceptable to construct the ontology for random sampling. Besides, the results in Section 5.7 demonstrate that our method can be implemented into generation-based backbone models.

Acknowledgements

The work was supported in part by NSFC with Grant No. 62293482, the Basic Research Project No. HZQB-KCZY-2021067 of Hetao Shenzhen-HK S&T Cooperation Zone, the National Key R&D Program of China with grant No. 2018YFB1800800, the Shenzhen Outstanding Talents Training Fund 202002, the Guangdong Research Projects No. 2017ZT07X152, No. 2019CX01X104, and No. 2021A1515011825, the Guangdong Provincial Key Laboratory of Future Networks of Intelligence (Grant No. 2022B12120100001), the Shenzhen Key Laboratory of Big Data and Artificial Intelligence (Grant No. ZDSYS201707251409055), and the National Key R&D Program of China under Grant No. 2020AAA0108600.

References


ACL 2023 Responsible NLP Checklist

A  For every submission:

- A1. Did you describe the limitations of your work?
  - *The final section after Conclusion*

- A2. Did you discuss any potential risks of your work?
  - *Not applicable. Left blank.*

- A3. Do the abstract and introduction summarize the paper’s main claims?
  - *Section 1*

- X A4. Have you used AI writing assistants when working on this paper?
  - *Left blank.*

B  ✓ Did you use or create scientific artifacts?

  - *Section 4*

- B1. Did you cite the creators of artifacts you used?
  - *Section 4*

- B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
  - *Not applicable. Left blank.*

- B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
  - *Not applicable. Left blank.*

- B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
  - *Not applicable. Left blank.*

- ✓ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
  - *Section 4*

- ✓ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
  - *Section 4*

C  ✓ Did you run computational experiments?

  - *Section 4*

- ✓ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
  - *Section 4*

---

*The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.*
C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
   No response.

C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
   No response.

C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
   No response.

D  X  Did you use human annotators (e.g., crowdworkers) or research with human participants?
   Left blank.

D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
   No response.

D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants’ demographic (e.g., country of residence)?
   No response.

D3. Did you discuss whether and how consent was obtained from people whose data you’re using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
   No response.

D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
   No response.

D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
   No response.