Abstract

Tracking dialogue states to better interpret user goals and feed downstream policy learning is a bottleneck in dialogue management. Common practice has been to treat it as a problem of classifying dialogue content into a set of pre-defined slot-value pairs, or generating values for different slots given the dialogue history. Both have limitations on considering dependencies that occur on dialogues, and are lacking of reasoning capabilities. This paper proposes to track dialogue states gradually with reasoning over dialogue turns with the help of the back-end data. Empirical results demonstrate that our method outperforms the state-of-the-art methods in terms of joint belief accuracy for MultiWOZ 2.1, a large-scale human–human dialogue dataset across multiple domains.

1 Introduction

Dialogue State Tracking (DST) usually works as a core component to monitor the user’s intentional states (or belief states) and is crucial for appropriate dialogue management. A state in DST typically consists of a set of dialogue acts and slot value pairs. Consider the task of restaurant reservation as shown in Figure 1. In each turn, the user may inform the agent of particular goals (e.g. single one as inform(food=Indian) or composed one as inform(area=center,food=Jamaican)). Such goals given during a turn are referred as turn belief. The joint belief is the set of accumulated turn goals updated until the current turn, which summarizes the information needed to successfully maintain and finish the dialogue.

Traditionally, dialogue system is supported by a domain ontology, which defines a collection of slots and the values that each slot can take. The aim of DST is to identify good features or patterns, and map to entries such as specific slot-value pairs in the ontology. It is often treated as a classification problem. Therefore, most efforts center on (1) finding salient features: from hand-crafted features (Wang and Lemon, 2013; Sun et al., 2014a), semantic dictionaries (Henderson et al., 2014b; Rastogi et al., 2017), to neural network extracted features (Mrkšić et al., 2017); or (2) investigating effective mappings: from rule-based models (Sun et al., 2014b), generative models (Thomson and Young, 2010; Williams and Young, 2007), to discriminative ones (Lee and Eskenazi, 2013; Ren et al., 2018; Xie et al., 2018). On the other hand, some researchers attack these methods’ over-dependence on domain ontology. They perform DST in the absence of a comprehensive domain ontology and handle unknown slot values by generating words from dialogue history or knowledge source (Rastogi et al., 2017; Xu and Hu, 2018; Wu et al., 2019).

However, the critical problem of modeling the dependencies and reasoning over dialogue history is not well researched. Many existing methods work on turn level only, which takes in the current turn utterance and outputs the corresponding turn belief (Henderson et al., 2014b; Zilka and Jurcicek, 2015; Rastogi et al., 2017; Xu and Hu, 2018). Compared to joint belief, the resulting turn belief only reflects single turn information, and thus is of less practical use. Therefore, more recent efforts target at the joint belief that summarizes the dialogue history. Generally speaking, they accumulate turn beliefs by rules ((Mrkšić et al., 2017; Zhong et al., 2018); Nouri and Hosseini-Asl, 2018) or model information across turns via various recurrent neural networks (RNNs) (Wen et al., 2017; Ramadan et al., 2018).
Although these RNN based methods model dialogue in turn by turn style, they usually feed the whole turn utterance directly to the RNN, which contains a large portion of noise, and result in unsatisfactory performance (Liao et al., 2018; Zhang et al., 2019b). More recently, there are works that directly merge fixed window of past turns (Perez and Liu, 2017; Wu et al., 2019) as new input and achieve state-of-the-art performance (Wu et al., 2019). Nonetheless, their capability of modeling long-range dependencies and doing reasoning in the interactive dialogue process is rather limited. For example, (Wu et al., 2019) performs gated copy to generate slot values from dialogue history. Although certain turns of utterances are exposed to the model, since the interactive signals are lost when concatenating turns together, it fails to do in-depth reasoning over turns.

Very recently, there is research starting to work in turn-by-turn style with pre-trained models. Generally speaking, such methods take the previous turn’s belief state and the current turn utterances as input to generate new dialogue state (Chao and Lane, 2019; Kim et al., 2020; Chen et al., 2020). However, there exists a long ignored fact that as an agent’s central component, the state tracker not only receives dialogue history but also observes the back-end database or knowledge base. Such an information source provides valuable hints for it to reason about user goals and update belief states. It is therefore natural to construct a bipartite graph based on the database where the entities and entity attributes are the two groups of nodes; with edges connecting them to express attribute belonging relation. As the example in Figure 1, the database does not contain restaurant entity serving Jamaican food and located in center area. Thus there would be no two-hop path between these two nodes. Existing methods like Wu et al. (2019) have to understand it via system utterances, while a DST reasoning over database would easily obtain such clues explicitly.

In this paper, we propose to do reasoning over turns and reasoning over database in Dialogue State Tracking (ReDST) for task-oriented systems. For reasoning over turns, we model dialogue state tracking as a recursive process in which the current joint belief relies on the generated current turn belief and last joint belief. Motivated by the limited length of single turn utterance and the good performance of pre-trained BERT (Devlin et al., 2019), we formalize the turn belief prediction as a token and sequence classification problem. It follows a multitask learning setting with augmented utterance inputs. To integrate the last turn belief results, an incremental inference module is applied for more robust belief updates. For reasoning over a database, we abstract the back-end database as a bipartite graph, and propagate extracted beliefs over the graph to obtain more realistic dialogue states. Contributions are summarized as:

- We propose to rethink the dialogue state tracking problem for task-oriented agents, pointing out the need for proper reasoning over turns and reasoning over back-end data.
- We represent the database into a bipartite graph and perform belief propagation on it, which enables the belief tracker to gain insight on potential candidates and detect conflicting requirements along the conversation course.
- With the help from pre-trained Transformer models working on augmented short utterance for achieving more accurate turn beliefs, we incrementally infer joint belief via reasoning in a turn by turn style and outperform state-of-the-art methods by a large margin.
2 Related Work

2.1 Dialogue State Tracking

A plethora of research has been focused on DST. We briefly discuss them in general chronological order. At the early stage, traditional dialogue state trackers combine semantic information extracted by Language Understanding (LU) modules to do DST (Williams and Young, 2007; Williams, 2014). Such trackers accumulate errors from the LU part and possibly suffer from information loss of dialogue context. Subsequent word-based (Henderson et al., 2014b; Zilka and Jurcicek, 2015) trackers thus forgo the LU part and directly infer states using dialogue history. Hand-crafted semantic dictionaries are utilized to hold all key terms, rephrases and alternative mentions to delexicalize for achieving generalization (Rastogi et al., 2017).

Recently, most approaches for dialogue state tracking rely on deep learning models (Wen et al., 2017; Ramadan et al., 2018). (Mrkšić et al., 2017) leveraged pre-trained word vectors to resolve lexical/morphological ambiguity. As it treats slots independently that might result in missing relations among slots (Ouyang et al., 2020), Zhong et al. (2018) proposed global modules to share parameters between estimators for different slots. Similarly, (Nouri and Hosseini-Asl 2018) used only one recurrent network with global conditioning to reduce latency while preserving performance. In general, these methods represent the dialogue state as a distribution over all candidate slot values that are defined in the ontology. This is often solved as a classification or matching problem. However, these methods rely heavily on a comprehensive ontology, which often might not be available. Therefore, Rastogi et al. (2017) introduced a sophisticated candidate generation strategy, while (Perez and Liu, 2017) followed the general paradigm of machine reading and proposed to solve it using an end-to-end memory network. Xu and Hu (2018) utilized the pointer network to extract slot values from utterances, while Wu et al. (2019) integrated copy mechanism to generate slot values.

However, these methods tend to largely ignore the dialogue logic and dependencies. For example, inter-utterance information and correlations between slot values have been shown to be challenging, let alone the frequent goal shifting of users. Consequently, reasoning over turns is sensible. We first aim to improve the turn belief prediction, then model the joint belief prediction as an updating process. Very recently, we see such design leveraged by several works. For example, Chao and Lane (2019) leverage BERT model to extract slot values for each turn, then employ a rule-based update mechanism to track dialogue states across turns. Ren et al. (2019) encode previous dialogue state and current turn utterances using Bi-LSTM, then hierarchically decode domains, slots, and values one after another. At the same time, Kim et al. (2020) encode these inputs with BERT model while predicting operation gates and generating possible values. Still, such methods largely ignore the fact that as an agent, it has access to the back-end data structure which can be leveraged to further improve the performance of DST.

2.2 Incremental Reasoning

The ability to do reasoning over the dialogue history is essential for dialogue state trackers. At the turn level, we aim to extract more accurate slot values from user utterance with the help of contextualized semantic inference. Contextualized representation learning in NLP dates back to Collobert and Weston (2008) but has had a resurgence in the recent year. Contextualized word vectors were pre-trained using machine translation data and transferred to text classification and QA tasks (McCann et al., 2017). Most recently, BERT (Devlin et al., 2019) employed Transformer layers (Vaswani et al., 2017) with a masked language modeling objective and achieved superior performance across various tasks. In DST, we also observe a wide adoption of such models (Shan et al., 2020; Liao et al., 2021). For example, Kim et al. (2020) and Heck et al., (2020) adopted the pre-trained BERT as base network. Hosseini-Asl et al. (2020) applied the pre-trained GPT-2 (Alec et al., 2019) model as the base network for dialogue state tracking.

At dialogue context level, since we perform reasoning via belief propagation through graph, our work is also related to a wide range of graph reasoning studies. As a relatively early work, the page-ranking algorithm (Page et al., 1999) used a random walk with restart mechanism to perform multi-hop reasoning. Almost at the same time, loopy belief propagation (Murphy et al., 1999) was proposed to calculate the approximate marginal probabilities of vertices in a graph based
on partial information. In recent years, research on graph reasoning has moved to learn symbolic inference rules from relational paths in the KG (Xiong et al., 2017; Das et al., 2017). Under these settings, a large number of entities and many types of relationships are usually involved. In DST, Chen et al. (2020) leveraged schema graphs containing slot relations, but their method heavily relied on a complete slot ontology. Zhou and Small (2019) incorporated a dynamically evolving knowledge graph to explicitly learn relationships slots. In our work, only the attribute-belonging relations are captured, and the constructed graph is simply a bipartite graph. We thus resort to heuristic belief propagation on the bipartite graph for reasoning. Further exploring more advanced models are treated as our future work.

### 3 ReDST Model

The proposed ReDST model in Figure 2 consists of three components: a turn belief generator, a bipartite graph belief propagator, and an incremental belief generator. Instead of predicting the joint belief directly from dialogue history, we perform two-stage inference: It first obtains turn belief from augmented turn utterance via transformer models. Then, it reasons over turn belief and last joint belief with the help of the bipartite graph propagation results. Based on this, it incrementally infers the final joint belief.

To facilitate the model description in detail, we first introduce our mathematical notations here. We define $X = \{(U_1, R_1), \ldots (U_T, R_T)\}$ as the set of user utterance and system response pairs in $T$ turns of dialogue, and $B = \{B_1, \ldots , B_T\}$ as the joint belief states at each turn. While $B_t$ summarizes the dialogue history up to the current turn $t$, we also model the turn belief $Q_t$ that corresponds to the belief state of a specific turn $(U_t, R_t)$, and denote $D_t$ as the domain of this specific turn. Following (Wu et al., 2019), we design our state tracker to handle multiple tasks. Thus, each $B_t$ or $Q_t$ consists of tuples like $(\text{domain}, \text{slot}, \text{value})$. Suppose there are $K$ different $(\text{domain}, \text{slot})$ pairs in total, we denote $Y_k$ as the true slot value for the $k$-th $(\text{domain}, \text{slot})$ pair.

#### 3.1 BERT-based Turn Belief Generator

Denoting $X_t = (U_t, R_t)$ as the $t$-th turn utterance, the goal of turn belief generator is to predict accurate state for this specific utterance. Although the dialogue history $X$ can accumulate in arbitrary length, the turn utterance $X_t$ is often relatively short in oftentimes. To utilize contextualized representation for extracting beliefs and enjoy the good performance of pre-trained encoders,
we fine-tune BERT as our base network while attaching the sequence classification and token classification layers in a multitask learning setting. The token classification task extracts specific slot value spans. The sequence classification task decides which domain the turn is talking about and whether a specific (domain, slot) pair takes the gate value like yes, no, don’t care, none, or generate from token classification, and so forth.

The model architecture of BERT is a multilayer bidirectional Transformer encoder based on the original Transformer model (Vaswani et al., 2017). The input representation is a concatenation of WordPiece embeddings (Wu et al., 2016), positional embeddings, and the segment embedding. As we need to predict the values for each (domain, slot) pair, we augment the input sequence as follows. Suppose we have the original utterance as \( X_t = x_1, \ldots, x_N \), the augmented utterance is then \( X'_t = [CLS], \text{domain}, \text{slot}, [SEP], x_1, \ldots, x_N, [SEP] \). The specific (domain, slot) works as queries to extract the answer span. We denote the outputs of BERT as \( H = h_1, \ldots, h_{N+5} \). The BERT model is pre-trained with two strategies on large-scale unlabeled text, that is, masked language model and next sentence prediction, which provide a powerful context-dependent sentence representation.

We use the hidden state \( h_t \) corresponding to [CLS] as the aggregated sequence representation to do the domain \( d^t \) and gate \( z^t \) classification:

\[
\begin{align*}
d^t &= \text{softmax}(W_{dn} \cdot (h_t) + b_{dn}), \\
z^t &= \text{softmax}(W_{gt} \cdot (h_t) + b_{gt})
\end{align*}
\]

where \( W_{dn} \) is trainable weight matrix and \( b_{dn} \) is the bias for domain classification. And \( W_{gt} \) is trainable weight matrix and \( b_{gt} \) is the bias for gate classification.

For token classification, we feed the hidden states of other tokens \( h_2, \ldots, h_{N+5} \) into a softmax layer to classify over the token labels \( S, I, O, [SEP] \) by

\[
y_n = \text{softmax}(W_{tc} \cdot (h_n) + b_{tc}),
\]

where \( W_{tc} \) is trainable weight matrix and \( b_{tc} \) is the bias for token classification.

To jointly model the sequence classification and token classification, we optimize their loss together. For the former, the cross-entropy loss \( L_{sc} \) is computed between the predicted \( d, z \) and the true one-hot label \( \hat{d}, \hat{z} \).

\[
L_{sc} = -\log(\hat{d} \cdot (d)^T) - \log(\hat{z} \cdot (z)^T).
\]

For the latter, we apply another cross-entropy loss \( L_{tc} \) between each token label in the input sequence.

\[
L_{tc} = -\sum_{n=2}^{N+5} \log(y_n \cdot (\hat{y}_n)^T).
\]

We optimize the turn belief generator via a weighted sum of these two loss functions as below over all training samples:

\[
L_{\text{turn}} = \alpha L_{sc} + \beta L_{tc}.
\]

### 3.1.1 Filter for Improving Efficiency

As in turn belief, most of the slots will get the value not mentioned. To enhance the efficiency of our model, we further design a gate mechanism similar to Wu et al. (2019) to filter out such slots first, for which we can skip the generation process and predict the value none directly. We apply the separate training objective as the cross-entropy loss computed between the predicted slot gate \( q^s_{\text{filter}} \) and the true one-hot label \( q^s_{\text{filter}} \) as below:

\[
L_{\text{filter}} = -\log(p^s_{\text{filter}} \cdot (q^s_{\text{filter}})^T),
\]

where for prediction, we calculate \( H_{X_t} = \mathcal{f}_{\text{BERT}}(X_t) \) as contextualized word representations for turn utterance, and then apply query attention to classify whether the slot should be filtered,

\[
\eta = \text{softmax}(H_{X_t} \cdot (q^s_{\text{filter}})^T),
\]

\[
p^s_{\text{filter}} = \text{softmax}(W_{\text{filter}} \cdot (\eta^T \cdot H_{X_t})^T).
\]

\( \text{W}_{\text{filter}} \) is the weight matrix and \( q^s_{\text{filter}} \) is the [CLS] position’s output from a BERT encoder for the domain-slot query.

### 3.2 Joint Belief Reasoning

Now we can predict the turn level belief state for each turn. Intuitively, we can directly apply our turn belief generator on concatenated dialogue history to obtain the joint belief as in Wu et al. (2019). However, it is hardly an optimal practice. First of all, treating all utterances as a long sequence will lose the
We use a bipartite graph for belief update. With reasoning over the agent’s database, infer such adjustment requirements from dialogue done. In most existing DSTs, the tracker has to adjust their requirements to get the task database and responds accordingly. Then, the user expresses requirements that, often, are hard to meet. The agent resorts to the back-end database to obtain credibility score for each slot value pairs. Then, we do incremental belief reasoning over the recursive process using different methods.

### 3.2.1 Bipartite Graph Belief Propagator

As the central component for dialogue systems, the dialogue state tracker has access to the back-end database most of the time. In the course of the task-oriented dialogue, the user and agent interact with each other to reach the same stage of information awareness regarding a specific task. The user expresses requirements that, often, are hard to meet. The agent resorts to the back-end database and responds accordingly. Then the user would adjust their requirements to get the task done. In most existing DSTs, the tracker has to infer such adjustment requirements from dialogue history. With reasoning over the agent’s database, we expect to harvest more accurate clues explicitly for belief update.

Consequently, we abstract the database as a bipartite graph $G = (V, E)$, where vertices are partitioned into two groups: The entity set $V_{ent}$ and attribute set $V_{attr}$, where $V = V_{ent} \cup V_{attr}$ and $V_{ent} \cap V_{attr} = \phi$. The entities within $V_{ent}$ and $V_{attr}$ are totally disconnected. Edges link two vertices from each of $V_{ent}$ and $V_{attr}$, representing the attribute belonging relationship. During each turn, we first map the predicted $Q_t$ and last joint belief $B_{t-1}$ to belief distributions over the graph via the function $g(\cdot)$. Here we apply fuzzy match and calculate the similarity with a threshold $\epsilon$ to realize $g(\cdot)$. We use BERT tokenizer to tokenize both dialogue and database entries. The mapping is done based on a preset threshold on the token level overlap ratio. For example, the generated ‘cambridge punt ##er’ will be mapped to the database entry ‘the cambridge punt ##er’ when their overlap ratio is larger than $\epsilon$. In our experiment, we find that approximately 60.5% of entity names and 12.2% other slot values can be mapped.\(^2\) This mapping operation actually helps to correct some minor errors made in span extraction or generation.

After the mapping of beliefs to the database bipartite graph via $g(\cdot)$, we start to do belief propagation over the graph. Generally speaking, there are two kinds of belief propagation in the bipartite graph. The first is from $V_{ent}$ to $V_{attr}$. It simulates the situation when a venue entity is mentioned, its attributes will be activated. For example, after a restaurant is recommended, a nearby hotel will have the same location value with it. The second one is from $V_{attr}$ to $V_{ent}$. This simulates the situation when an attribute is mentioned, all entities having this attribute will also receive the propagated beliefs. If an entity gets more attributes mentioned, it will receive more propagated beliefs. Suppose the propagation result is $c_t$ for the current turn $t$, it can be viewed as the credibility scores of the state values after reasoning over the database graph. We reason over this set of entries via doing belief propagation in the bipartite graph to obtain the certainty scores for them as below:

$$c_t = \gamma \cdot g(B_{t-1}) + \eta \cdot g(Q_t) \cdot (I + W_{adj}),$$

where $\gamma$ is a hyper-parameter for modeling the credibility decay, because newly provided slot values usually reflect more updated user intention. $\eta$ adjusts the effect of propagated beliefs. $W_{adj}$ is the adjacency matrix of the bipartite graph. Note that the belief propagation method is rather simple but effective. We tried more advanced methods such as loopy belief propagation (Murphy et al., 1999). However, we did not see obvious performance gain, which might be due to the relatively small bipartite graph size (273 nodes in total). Also, we suspect that graph reasoning might be more helpful for downstream tasks such as action prediction. We will explore further in future.

### 3.2.2 Incremental Belief Generator

With the incremental belief generator, we now incrementally infer the...

\(^2\)Over half of the slot values are time, people, stay, day, etc. There are no such nodes in the bipartite graph but we keep these slot values’ existence in the belief vector.
current joint belief \( B_t \). Mathematically, we have
\[
B_t = f(Q_t, B_{t-1}, c_t). \tag{6}
\]
The function \( f \) integrates evidence from the turn belief, last joint belief, and the propagated credibility scores. There are wide variety of models that can be applied. We may leverage the straight-forward Multi-Layer Perceptron (MLP) to model the interactions between these beliefs (He et al., 2017) deeply. Due to the sequential nature of the belief generator, we can also apply GRU cells to predict the beliefs turn by turn (Cho et al., 2014). Intuitively, given these remaining and new belief entries as well as credibility scores, the essential task here is to reason out what entries to keep, update, or delete. Therefore, we make use of these information to carry out the operation classification task. There are three operations keep, update, and delete to choose from for each domain slot. For the GRU case, the detailed equation for operation classification is as below:
\[
\begin{align*}
    h_t &= GRU(W \cdot [g(Q_t), c_t], h_{t-1}) \\
    op_k &= softmax(W_{op_k} \cdot (h_t)^T + b_{op_k}),
\end{align*}
\]
where \( W \cdot [g(Q_t), c_t] \) and \( h_{t-1} \) are the inputs to the GRU cell. \([,]\) denotes vector concatenation. \( W_{op_k} \) and \( b_{op_k} \) are the weight matrix and bias vector for the corresponding \( k \)-th \((\text{domain, slot})\) pair. After the operation \( \text{op} \) in the current turn \( t \) is predicted, we obtain the corresponding current joint belief \( B_t \) via performing corresponding operations.

4 Experiments

4.1 Dataset

We carry out experiments on MultiWOZ 2.1 (Eric et al., 2019). It is a multi-domain dialogue dataset spanning seven distinct domains and containing over 10,000 dialogues. As compared to MultiWOZ 2.0, it fixed substantial noisy dialogue state annotations and dialogue utterances that could negatively impact the performance of state-tracking models. In MultiWOZ 2.1, there are 30 domain-slot pairs and over 4,500 possible values, which is different from existing standard datasets like WOZ (Wen et al., 2017) and DSTC2 (Henderson et al., 2014a), which have fewer than ten slots and only a few hundred values. We follow the original training, validation, and testing split and directly use the DST labels. Since the hospital and police domain have very few dialogues (10% compared to others) and only appear in the training set, we only use the other five domains in our experiment.

4.2 Settings

Training Details Our model is trained in a two-stage style. We first train the turn belief generator using the Adam optimizer with a batch size of 32. We adopt the bert-base-uncased version of BERT and initialize the learning rate for fine-tuning as 3e-5. The \( \alpha \) and \( \beta \) in Equation 4 are set to 0.05 and 1.0, respectively. We use the average of the last four hidden layer outputs of BERT as the final representation of each token.

During the later reasoning stage, regarding incremental belief reasoning, we use a fully connected two-layer feed-forward neural network with ReLU activation for MLP. The hidden size is set to 500, and the learning rate is initialized as 0.002. For GRU, we set the learning rate as 0.005. We pre-process turn utterances to alleviate the problem of ground truth absence, for example, formalize time values into standard forms. Similar to Heck et al. (2020), we also make use of the system acts to enrich the system utterances.

Evaluation Metrics Similar to Wu et al. (2019), we adopt the evaluation metric joint goal accuracy to evaluate the performance. It is a relatively strict evaluation standard. The joint goal accuracy compares the predicted belief states to the ground truth \( B_t \) at each turn \( t \). The joint accuracy is 1.0 if and only if all \((\text{domain, slot, value})\) triplets are predicted correctly at each turn, otherwise it is 0.

Baselines We denote the two versions of ReDST with different incremental reasoning modules as ReDST\(_{MLP}\), and ReDST\(_{GRU}\). They are compared with the following baselines.

DST Reader (Gao et al., 2019): It treats DST as a reading comprehension problem. Given the history, it learns to extract slot values as spans.

HyST (Goel et al., 2019): It combines a hierarchical encoder in a fixed vocabulary system with an open vocabulary n-gram copy-based system.

TRADE (Wu et al., 2019): It concatenates the whole dialogue history as input and uses a generative state tracker with a copy mechanism to generate value for each slot separately.
**DST-Picklist** (Zhang et al., 2019a): Given the whole dialogue history as input, it uses two BERT-based encoders and takes a hybrid approach of predefined ontology-based DST and open vocabulary-based DST. It defines picklist-based slots for classification and span-based slots for span extraction like DSTRead (Gao et al., 2019).

**SOM** (Kim et al., 2020): It works in turn-by-turn style and considers state as an explicit fixed-sized memory, and adopts a selectively overwriting mechanism for generating values with copy.

**SST** (Chen et al., 2020): It leverages a graph attention matching network to fuse information from utterances and schema graphs. A recurrent graph attention network controls state updating. It relies on a predefined ontology.

### 4.3 DST Results

We first compare our model with the state-of-the-art methods. As shown in Table 1, we observe that our method outperforms all the other baselines. For example, in terms of joint accuracy, which is a rather strict metric, ReDST$_{GRU}$ improves the performance by 46.2%, 17.4%, and 1.3% as compared to open-vocabulary based methods: the DST Reader, TRADE, and SOM, respectively. Based on results in Table 1, the methods such as DST-Picklist and SST perform better than our method. However, they rely heavily on a predefined ontology. In such methods, the value candidates for each slot to choose from are fixed already. They cannot handle unknown slot values, which largely limits their application in real-life scenarios.

We observe that a large portion of baselines work on relatively long window-sized dialogue history. FJST directly encodes the raw dialogue history using recurrent neural networks. In contrast, HJST first encodes turn utterance to vectors using a word-level RNN, and then encodes the whole history to vectors using a context level RNN. However, the lower performance of HJST demonstrates its inefficiency in learning useful features in this task. Based on HJST, HyST manages to achieve better performance by further integrating a copy-based module. Still, the performance is lower than TRADE, which encodes the raw concatenated whole dialogue history, generates or copies slot values with extra slot gates. Generally speaking, these baselines are based on recurrent neural networks for encoding dialogue history. Since the interactions between user and agent can be arbitrarily long and recurrent neural networks are not effective in modeling long-range dependencies, they might not be a good choice to model the dialogue for DST. On the contrary, single turn utterances usually are short and contain relatively simple information as compared to complicated dialogue history. It is thus better to generate belief in turn level and then integrate them via reasoning. According to the comparisons of baselines, the superior performance of SST, SOM, and ReDSTs validate this design.

Moreover, we also tested the performance of TRADE without the slot gate. The performance drops dramatically—from 0.453 to 0.411 in terms of joint accuracy. We suspect that this is due to lengthy dialogue history, where the decoder and copy mechanism start to lose focus. It might generate some value that appears in dialogue history but is not the ground truth. Therefore, the slot gate is used to decide which slot value should be taken, which resembles the inference in some sense. To validate this, we feed the single turn utterances to TRADE and generate the turn beliefs as output. Interestingly, we find that it performs similar with gate or without it, which validates our guess. However, such resembled inference is not enough. When the dialogue history becomes long, the gating mechanism will lose its focus easily. Accordingly, we report the results of TRADE and ReDST$_{GRU}$ on the last four turns of dialogues in Table 2. The better performance of ReDST$_{GRU}$

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>DST-Picklist</td>
<td>0.533</td>
</tr>
<tr>
<td>SST</td>
<td>0.552</td>
</tr>
<tr>
<td>DST Reader</td>
<td>0.364</td>
</tr>
<tr>
<td>TRADE</td>
<td>0.453</td>
</tr>
<tr>
<td>TRADE w/o gate</td>
<td>0.411</td>
</tr>
<tr>
<td>SOM</td>
<td>0.525</td>
</tr>
<tr>
<td>ReDST$_{MLP}$</td>
<td>0.511</td>
</tr>
<tr>
<td>ReDST$_{GRU}$</td>
<td>0.532</td>
</tr>
</tbody>
</table>

Table 1: The multi-domain DST evaluation results on the MultiWOZ 2.1 dataset. The ReDST$_{GRU}$ method achieves the highest joint accuracy.
Table 2: The last four turns’ joint accuracy of TRADE and proposed ReDST. (T refers to the last turn of each dialogue session.)

<table>
<thead>
<tr>
<th>Model</th>
<th>T-3</th>
<th>T-2</th>
<th>T-1</th>
<th>T</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADE</td>
<td>0.411</td>
<td>0.339</td>
<td>0.269</td>
<td>0.282</td>
</tr>
<tr>
<td>ReDST_GRU</td>
<td>0.487</td>
<td>0.440</td>
<td>0.391</td>
<td>0.377</td>
</tr>
</tbody>
</table>

Table 3: The turn belief generation results of TRADE, SOM, and proposed ReDST.

<table>
<thead>
<tr>
<th>Model</th>
<th>Joint Acc</th>
</tr>
</thead>
<tbody>
<tr>
<td>TRADE</td>
<td>0.697</td>
</tr>
<tr>
<td>SOM</td>
<td>0.799</td>
</tr>
<tr>
<td>ReDST</td>
<td>0.808</td>
</tr>
</tbody>
</table>

Table 4: The joint accuracy results for ReDST methods with or without bipartite graph reasoning.

<table>
<thead>
<tr>
<th>Setting</th>
<th>w BP</th>
<th>w/o BP</th>
</tr>
</thead>
<tbody>
<tr>
<td>ReDST_MLP</td>
<td>0.511</td>
<td>0.507</td>
</tr>
<tr>
<td>ReDST_GRU</td>
<td>0.532</td>
<td>0.530</td>
</tr>
</tbody>
</table>

4.4 Component Analysis

Since our model makes use of the advanced BERT structure to learn the contextualized representation, we first test how much contribution the BERT has made. Therefore, we carried out a study on a turn belief generator and compare it with SOM and the BiLSTM baseline TRADE on the single turn utterance. As shown in Table 3, we observe that the BERT-based SOM and ReDST indeed perform better than single turn TRADE. This is due to the usage of pre-trained BERT in learning better-contextualized features. In the multitask setting of our design, both the token classification and sequence classification tasks benefit from BERT’s strength. Moreover, we notice that when doing the single turn setting, the system response usually depends on certain information mentioned in the former turn user utterance. Therefore, we concatenate the former turn utterance to each current single turn as the input for BERT. Under this setting, we achieved a large boost in performance regarding joint accuracy as in Table 3. It provides an excellent base for the later stage inferences.

We also tested the effect of reasoning over the database. For a clear comparison, we ignore the evidence obtained via bipartite graph belief propagation while keeping other settings the same. To show it more clearly, we re-organize the results in Table 4. It can be observed that both ReDST\_MLP and ReDST\_GRU gain a bit from belief propagation. It validates the usefulness of database reasoning. However, since the graph is rather small, the performance improvement is rather limited. Similar patterns are found in Chen et al. (2020) and we suspect that it will be more helpful with larger database structure. Also, we will further explore its usage in downstream tasks such as action prediction.

For different incremental reasoning modules, the results are also shown in Table 1. We find that ReDST\_GRU performs better. However, we notice that simply accumulating turn belief as in Zhong et al. (2018) performs very well. The rule is to add newly predicted turn belief entries to the last joint belief. When different values for a slot appear, only keep the new one. Although this rule seems simple, it actually reflects the dialogue’s interactive and updating nature. We tried to directly apply this rule on the ground truth turn belief to generate joint belief. It results in 0.963 joint accuracy. However, a critical problem of such accumulation rule is that when the generated turn belief is wrong, it will not be able to add a missing entry or delete a wrong entry. By applying GRU in ReDST\_GRU, it manages to modify a bit with the help of database evidence. Still, there is large space for more powerful reasoning models to address this error accumulation issue. We will further investigate in this direction.

4.5 Error Analysis

We also provide error analysis regarding each slot for ReDST\_GRU in Figure 3. To make it more clear, we also list the results of SOM for comparison. We observe that a large portion of the improvements for our method are on name entities and time-related slots. As mentioned in Wu et al. (2019), name slots in the attraction, restaurant, and hotel domains have the highest error rates. It is partly because these slots have a relatively
Figure 3: Slot error rate on the test set. The error rate for name slots on restaurant, hotel, and attraction domain drops 4.2% on average.

A large number of possible values that are hard to recognize. In ReDSTGRU, we map beliefs into a bipartite graph constructed via database and do belief propagation on it. This helps to improve the accuracy on name slots. Also, the classification gate design helps to improve performance on Yes/No slots. We also observe that the performance for taxi destination becomes worse. This is due to the value co-reference phenomenon where the user might just mention ‘taxi to the hotel’ to refer to the hotel name mentioned earlier. These findings are interesting and we will explore it further.

5 Conclusion

We rethink DST from the angle of agent and point out the urgent need for in-depth reasoning other than being obsessed with generating values from history text as a whole. We demonstrated the importance of doing reasoning over turns and over the database. In detail, we fine-tuned pre-trained BERT for more accurate turn level belief generation while doing belief propagation in bipartite graph to harvest more clues. Experiments on a large-scale multi-domain dataset demonstrate the superior performance of the proposed method. In the future, we will explore more advanced algorithms for performing reasoning over turns and on graphs for generating more accurate summarization of user intention.

Acknowledgments

This research is supported by the National Research Foundation, Singapore, under its International Research Centres in Singapore Funding Initiative. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not reflect the views of National Research Foundation, Singapore.

References


Rajarshi Das, Shehzaad Dhulliawala, Manzil Zaheer, Luke Vilnis, Ishan Durugkar, Akshay Krishnamurthy, Alex Smola, and Andrew McCallum. 2017. Go for a walk and arrive at the answer: Reasoning over paths in knowledge


Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google’s neural machine translation system: Bridging the
gap between human and machine translation.  


