arXiv:2005.13129v1 [cs.AI] 27 May 2020

Rethinking Dialogue State Tracking with Reasoning

Lizi Liao, Yunshan Ma, Wenqiang Lei, Tat-Seng Chua
School of Computing
National University of Singapore
{liaolizi.llz, yunshan.ma, wenqianglei}@gmail.com
chuats@comp.nus.edu.sg

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 significantly outperforms the state-of-the-art methods by 38.6% 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, a dialogue system is supported by a *domain ontology*, which defines a collection of slots and the values that each slot can take. DST aims 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 criticize the over-dependence of these methods on domain ontology; instead, 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).

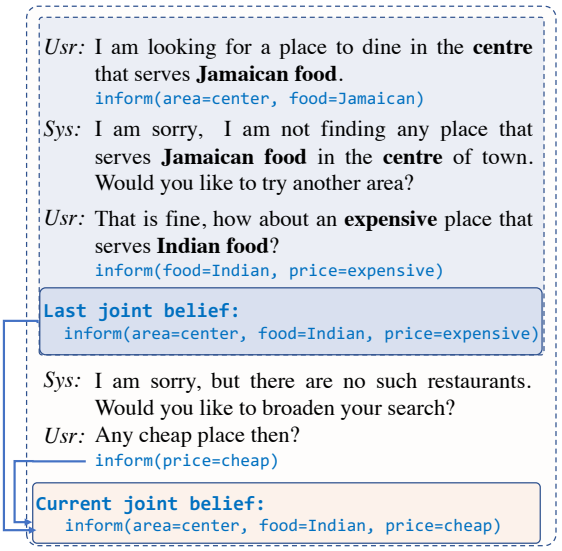

Figure 1: An example dialogue for illustration. Each turn contains a system response and a user utterance. Turn belief labels are provided based on turn information, while the joint belief captures most updated user intention up to the current turn.

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 output 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 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 (RNN) (Wen et al., 2017; Ramadan et al., 2018; Lei 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. 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.

Moreover, 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 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 [1]) 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, and an annealing training strategy further mitigates the gap between training and testing phases. For reasoning over 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 belief tracker to gain insight on potential candidates and detect conflicting requirements along the conversation course.
- With the help from pre-trained BERT 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.

In what follows, we will summarize the related efforts to our work in Section 2, describe the proposed ReDST method in Section 3, and provide detailed experimental results in Section 4.

2 Related Work

2.1 Dialogue State Tracking

A plethora of research has been focused on dialogue state tracking. We briefly discuss them in general chronological order.

At early stage, traditional dialogue state trackers combine semantic information extracted by Language Understanding (LU) modules to estimate

[1] https://github.com/lizi-git/ReDST

the current dialogue states (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 the state using dialogue history. Handcrafted semantic dictionaries are utilized to hold all key terms, rephrasing and alternative mentions to *delexicalize* for achieving generalization (Rastogi et al., 2017).

Recently, most state-of-the-art approaches for dialogue state tracking rely on deep learning models (Wen et al., 2017; Lei et al., 2018; Ramadan et al., 2018). (Mrkšić et al., 2017) leveraged pre-trained word vectors to resolve lexical/morphological ambiguity. As it treats slots independently, which might result in issues in training and missing relations among slots, (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. It 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. Moreover, (Wu et al., 2019) integrated copy mechanism to generate slot values in states.

More importantly, current methods tend to fail at considering dependencies that occur in dialogues. For example, inter-utterance information and correlations between slot values have been shown to be challenging to handle, let alone the frequent goal shifting of users. Consequently, reasoning over turns becomes an essential requirement for DST. We thus first aim to improve the turn belief prediction, then model the joint belief prediction as a recursive process. Furthermore, current 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. Thus, in this work, we propose to leverage the back-end database to help DST.

2.2 Incremental Reasoning

The ability to do reasoning over the dialogue history is essential for dialogue state tracker to find user intention. At the turn level, we aim to extract more accurate slot values from user utterance with the help of contextualized semantic inference. In this direction, our work is related to the use of pre-trained encoders for contextualized representation learning in NLP, which 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). ELMO improved contextualized word vectors by using a language modeling objective (Peters et al., 2018). 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.

At dialogue context level, DST should perform reasoning over turns where relations between slot values across turns need to be captured, and hints from the database should also be leveraged. As we perform reasoning via belief propagation through graph based on the database, 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 and being formulated as sequential decision problems (Xiong et al., 2017; Das et al., 2017). Under these settings, a large number of entities and many types of relationships are usually involved. However, 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 single turn utterance via BERT token and sequence classification. 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 $X = \{(U_1, R_1), \cdots (U_T, R_T)\}$ as the set of user utterance and system response pairs in T turns of dialogue, and $B = \{B_1, \cdots, B_T\}$ 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). 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 $(domain, slot, value)$. Suppose there are K different $(domain, slot)$ pairs in total, Y_k is the true slot value for the k-th $(domain, slot)$ pair.

3.1 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, and the sequence classification task decides whether a specific $(domain, slot)$ pair takes the value like $yes, no, doncare, none$, or $generate$ from token classification *etc*.

The model architecture of BERT is a multi-layer 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 $X_t = x_1, \cdots, x_N$, the augmented utterance is then X'_t = [CLS], $domain$, $slot$, [SEP], $x_1, \cdots, x_N$, [SEP]. The specific $(domain, slot)$ works as queries to extract the answer span in utterance. We denote the outputs of BERT as $H = \mathbf{h}_1, ..., \mathbf{h}_{N+5}$ [2]. The BERT model is pre-trained with two strategies on large-scale unlabeled text, *i.e.*, masked language model and next sentence prediction, which provide a powerful context-dependent sentence representation.

We use the hidden state $\mathbf{h}_1$ corresponding to [CLS] as the aggregated sequence representation to do the classification:

$$\mathbf{z}^k = softmax(\mathbf{W}^k_{sc} \cdot (\mathbf{h}_1)^T + \mathbf{b}^k_{sc}), \quad (1)$$

where $\mathbf{W}^k_{sc}$ is trainable weight matrix and $\mathbf{b}^k_{sc}$ is the bias for sequence classification.

For token classification, we feed the hidden states of other tokens $\mathbf{h}_2, \cdots, \mathbf{h}_{N+5}$ into a softmax layer to classify over the token labels S, I, O, [SEP] by

$$\mathbf{y}^k_n = softmax(\mathbf{W}^k_{tc} \cdot (\mathbf{h}_n)^T + \mathbf{b}^k_{tc}), \quad (2)$$

where $\mathbf{W}^k_{tc}$ is trainable weight matrix and $\mathbf{b}^k_{tc}$ is the bias for token classification.

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

$$L_{sc} = -\sum_{k=1}^{K} log(\mathbf{z}^k \cdot (\hat{\mathbf{z}}^k)^T). \quad (3)$$

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

$$L_{tc} = -\sum_{k=1}^{K} \sum_{n=2}^{N+5} log(\mathbf{y}^k_n \cdot (\hat{\mathbf{y}}^k_n)^T). \quad (4)$$

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

$$L_{turn} = \alpha L_{sc} + \beta L_{tc}. \quad (5)$$

[2] For ease of illustration, we ignore the WordPiece separation effect on token numbers.

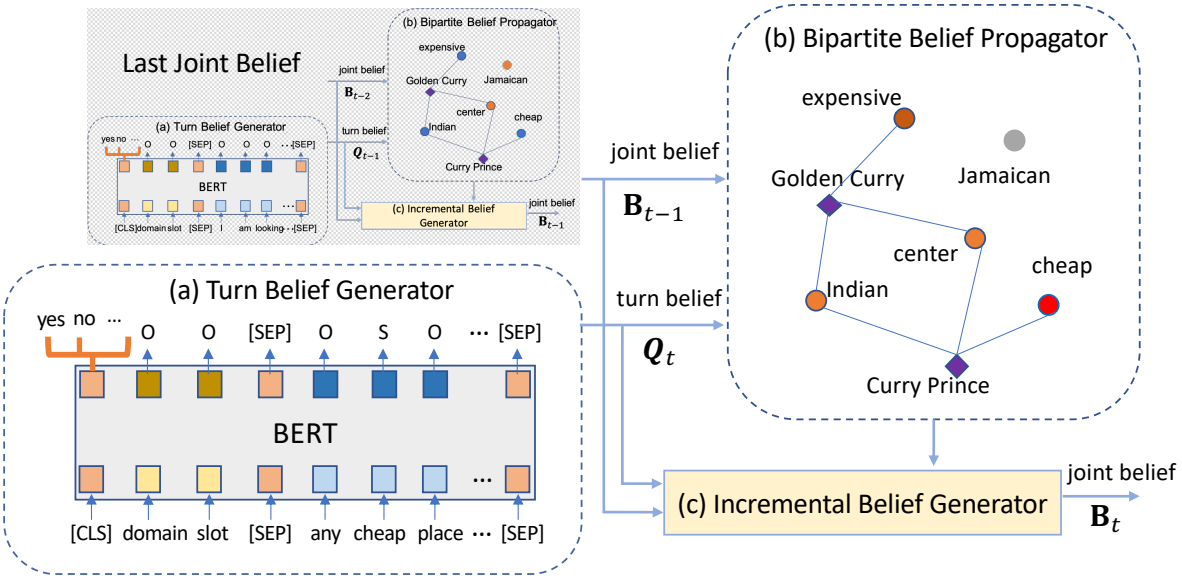

Figure 2: The architecture of the proposed ReDST model, which comprises (a) a turn belief generator, (b) a bipartite belief propagator, and (c) an incremental belief generator. The turn belief generator will predict values for all domain slot pairs. Together with the last joint belief, the beliefs will be aggregated via the bipartite belief propagator based on the database structure. Then the incremental belief generator infers the final joint belief.

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 iterative character of dialogue, thus resulting in information loss. Secondly, current models like recurrent networks or Transformers are known for not being able to model the long-range dependencies well. Therefore, we simulate the dialogue procedure as a recursive process where current joint belief B_t relies on last joint belief B_{t-1} and the current turn belief Q_t. Generally speaking, we use B_{t-1} and Q_t to perform belief propagation on the Bipartite graph constructed based on 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. In the course of the task-oriented dialogue, the user and agent interact with each other to reach the same stage of information awareness. The user expresses requirements that, many times, are hard to meet. The agent resorts to the back-end database and responds accordingly. Then the user would adjust his/her 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.

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 exact match and calculate the similarity with a threshold ϵ to realize $g(\cdot)$. We use BERT tokenizer to tokenize both dialogue and database entries. The mapping is done based on a pre-set 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 ϵ. In our experiment, we find that approximately 60.5% of entity names and 12.2% other slot values can be mapped [3].

[3] Over half of the slot values are time, people, stay, day *etc.*. We ignore these as no entity links to them.

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 $\mathbf{c}_t$ 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:

$$\mathbf{c}_t = \gamma \cdot g(B_{t-1}) + \eta \cdot g(Q_t) \cdot (\mathbf{I} + \mathbf{W}^{adj}), \quad (6)$$

where γ is a hyper-parameter for modeling the credibility decay, because newly provided slot values usually reflect more updated user intention. η adjusts the effect of propagated beliefs. $\mathbf{W}^{adj}$ 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 database size. There are only 273 nodes in the bipartite graph in total.

3.2.2 Incremental Belief Generator

With the credibility scores $\mathbf{c}_t$ obtained from the belief propagator, we now incrementally infer the current joint belief B_t. Mathematically, we have

$$B_t = f(Q_t, B_{t-1}, \mathbf{c}_t). \quad (7)$$

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. For example, we can apply simple rules to merge these beliefs as in (Zhong et al., 2018), which directly accumulates beliefs by adding or updating slot values. 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 a GRU cell to predict the beliefs turn by turn (Cho et al., 2014). For GRU case, the detailed equation is as below:

$$g(B_t) = GRU(\mathbf{W} \cdot [g(Q_t); \mathbf{c}_t], g(B_{t-1})), \quad (8)$$

where $\mathbf{W} \cdot [g(Q_t); \mathbf{c}_t]$ and $g(B_{t-1})$ are the inputs to the GRU cell. $[;]$ denotes vector concatenation. We carry out experiments on different models, compare and analyze their performance.

3.3 Optimization Strategy

As dialogue goes on, our model sequentially generates joint belief in the context of last joint belief. At training time, it can predict from the ground truth of last turn belief, while at inference, it has to generate the sequence of beliefs from scratch. This discrepancy of feeding the last joint belief leads to error accumulation among the way. Inspired from (Zhang et al., 2019), we sample the last joint belief not only from the ground truth $\hat{B}_{t-1}$ but also from the predicted one B_{t-1} by the model during training. At the beginning of training, as the model is not well trained, using the predicted B_{t-1} as input would often lead to very slow convergence, even being trapped in local optimum. Thus, borrowing the idea from (Zhang et al., 2019), we define the probability p of selecting from the ground truth with a decay function dependent on the training epoch number e:

$$p = \frac{\mu}{\mu + exp(e/\mu)} \quad (9)$$

where μ is a hyper-parameter. The value of p will decrease constantly which corresponds to the probability of feeding the ground truth in the training process.

4 Experiments

4.1 Dataset

We carry out experiments on MultiWOZ 2.1 (Eric et al., 2019), which is a recently released multi-domain dialogue dataset spanning seven distinct domains and containing over 10,000 dialogues. As compared to the old version 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

less 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 α and η in Equation 5 are set to one and ten 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, we set the credibility decay γ for last joint belief as 0.5 and the propagation effect adjustments η as 1.0. 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 use it with a hidden size of 500 and learning rate of 0.005.

Evaluation Metrics Similar to (Wu et al., 2019), we adopt two evaluation metrics, joint goal accuracy and slot accuracy, to evaluate the performance of multi-domain DST. 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 $(domain, slot, value)$ triplets are predicted correctly at each turn, otherwise 0. The slot accuracy, on the other hand, individually compares each $(domain, slot, value)$ triplet to its ground truth label.

Baselines We denote the three versions of ReDST with different incremental reasoning modules as ReDST$_{_RULE}$, ReDST$_{_MLP}$, and ReDST$_{_GRU}$. They are compared with the following baselines. More details about these methods are given below:

- FJST: It refers to a bidirectional LSTM network that encodes the full dialogue history and then applies a separate feedforward network to the encoded hidden state for every single state slot.
- HJST (Serban et al., 2016): It also considers the full dialogue history similar to FJST but instead encodes it using a hierarchical recurrent neural network following the tradition of (Serban et al., 2016).
- DST Reader (Gao et al., 2019): It is a newly proposed model that treats dialogue state tracking as a reading comprehension problem. Given the dialogue history, it learns to extract slot values as spans.
- HyST (Goel et al., 2019): This is another new model that combines a hierarchical encoder in a fixed vocabulary system with an open vocabulary n-gram copy-based system.
- TRADE (Wu et al., 2019): This is the current state-of-the-art model on the multi-domain MultiWOZ 2.1 dataset. It concatenates dialogue history as input and uses a generative state tracker with a copy mechanism.

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 with a large margin. For example, in terms of joint accuracy which is a rather strict metric, ReDST$_{_GRU}$ improves the performance by $66.1\%, 76.4\%, 72.5\%, 64.8\%$, and 38.6% as compared to the FJST, HJST, DST Reader, HyST, and TRADE, respectively.

All these baselines work on 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 the current state-of-the-art system TRADE, which encodes the raw dialogue history, generates or copies slot values with extra slot gates.

Generally speaking, all 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,

Model	Joint Acc	Slot Acc
FJST	0.378	0.952
HJST	0.356	0.955
DST Reader	0.364	0.952
HyST	0.381	0.961
TRADE	0.453	0.970
TRADE w/o gate	0.411	0.960
ReDST$_{_RULE}$	0.400	0.958
ReDST$_{_MLP}$	0.552	0.974
ReDST$_{_GRU}$	**0.628**	**0.983**

Table 1: The multi-domain DST evaluation results on the MultiWOZ 2.1 dataset. The proposed ReDST$_{_GRU}$ method achieves the highest joint accuracy, which surpasses the current state-of-the-art TRADE model by a large margin.

Model	T-3	T-2	T-1	T
TRADE	0.411	0.339	0.269	0.282
ReDST	**0.607**	**0.519**	**0.489**	**0.672**

Table 2: The last four turns' joint accuracy of TRADE and proposed ReDST$_{_GRU}$. (T refers to the last turn of each dialogue session.)

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. Compared to baselines, the superior performance of various ReDSTs generally validate our 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 fall short of hands. 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$_{_GRU}$

Model	Joint Acc	Slot Acc
TRADE	0.697	0.986
ReDST	**0.726**	**0.987**

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

Setting	w BP	w/o BP
ReDST$_{_RULE}$	N/A	0.400
ReDST$_{_MLP}$	0.546	**0.552**
ReDST$_{_GRU}$	**0.628**	0.607

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

further validates the importance of reasoning. Usually, as the interactive dialogue goes on, users might frequently adjust their goals, which requires special consideration. Since turn utterance is relatively more straightforward and dialogue is turn by turn in nature, doing DST in a turn by turn style is a useful and practical design.

4.4 Component Analysis

Since our model makes use of the advanced BERT structure to learn the contextualized representation, one might suspect whether the superior performance comes from BERT. Therefore, we carried out study on turn belief generator which does not involve the reasoning components. We compare it with the best performance baseline TRADE on the single turn utterance. The results are shown in Table 3. We observe that ReDST performs better than single turn TRADE. This is partially 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. It provides an excellent base for the later stage inferences. However, we notice that the performance gain is rather limited — only 4.2% in terms of joint accuracy. Therefore, the large performance gain of ReDST$_{_GRU}$ over other baselines comes from doing reasoning. This is supported by the observation that ReDST$_{_RULE}$ (using BERT but without strong reasoning in multiple turn setting) performs much worse than TRADE as shown in Table 1.

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 clear, we re-organize the results in Table 4. Since the database reasoning results cannot be integrated without changing our integration rules, we did not integrate it into our rule-based method. It can be observed that ReDST$_{_GRU}$ gains the most with belief propagation. It validates the usefulness of database reasoning. However, for ReDST$_{_MLP}$, the result actually gets worse, which implies that finding a good way to integrate such information is the key.

For different incremental reasoning modules, the results are also shown in Table 1. We find that ReDST$_{_GRU}$ performs the best. Simply accumulating turn belief as in (Zhong et al., 2018) performs the worst, which also reflects the need for doing reasoning. Using MLP to capture the interactions among the last joint beliefs and turn belief yields better results. However, its performance is still far from using GRU units. Intuitively, GRU units have the gating mechanism which can adaptively remember and forget information. In our DST setting, certain information in the last joint belief will be disposed of or updated while the other information will be carried onto future turns, which naturally fits the recurrent modeling paradigm.

4.5 Error Analysis

We also provide error analysis regarding each slot for ReDST$_{_GRU}$ and compare it with that of TRADE as shown in Figure 3. We observe that most of the improvements of our method are on name entities. 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 usually have a large number of possible values that are hard to recognize. Another important reason is that their model lacks reasoning on the relationship between the entity (value of $name$ slots) and attributes (value of some other slots). However, ReDST$_{_GRU}$ manages to achieve much lower error rates on these $name$ slots. This is because we map beliefs into bipartite graph constructed via database and do belief propagation on it. In this way, we directly capture the relationships and do reasoning among them. In addition, since relations between nodes are considered, we observe that slots like $type$ and $area$ are also largely improved which might due to their close connections to named entities. However, the time related slots did not improve much. This could be due to their loose connections to other entities.

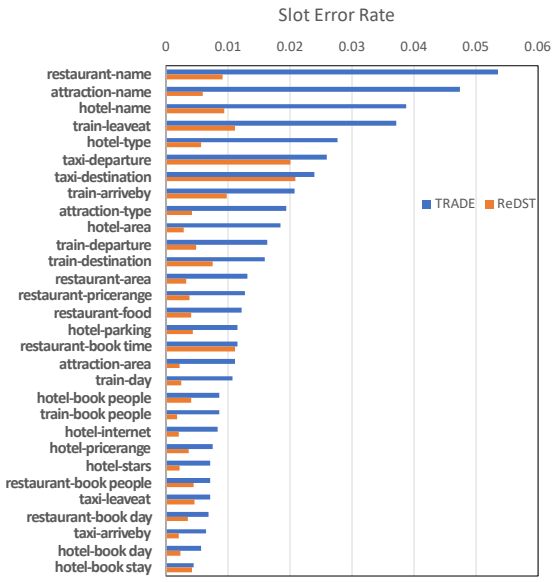

Figure 3: Slot error rate on the test set. The error rate for $name$ slots on $restaurant, hotel$ and $attraction$ domain drops 82.02% on average.

5 Conclusion

We rethink the dialogue state tracking problem 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 doing reasoning 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. Different models are applied to perform incremental reasoning to generate joint beliefs. 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, investigate the more detailed relationships between nodes, and examine other possibilities of doing incremental reasoning for more accurate summarization of user intention.

Acknowledgment

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *EMNLP*, pages 1724–1734.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *ICML*, pages 160–167.

Rajarshi Das, Shehzaad Dhuliawala, 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 bases using reinforcement learning. *arXiv preprint arXiv:1711.05851*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186.

Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyang Gao, and Dilek Hakkani-Tür. 2019. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *CoRR*, abs/1907.01669.

Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog state tracking: A neural reading comprehension approach. In *SIGDIAL*, pages 264–273.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. *arXiv preprint arXiv:1907.00883*.

Xiangnan He, Lizi Liao, Hanwang Zhang, Liqiang Nie, Xia Hu, and Tat-Seng Chua. 2017. Neural collaborative filtering. In *WWW*, pages 173–182. International World Wide Web Conferences Steering Committee.

Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014a. The second dialog state tracking challenge. In *SIGDIAL*, pages 263–272.

Matthew Henderson, Blaise Thomson, and Steve Young. 2014b. Word-based dialog state tracking with recurrent neural networks. In *SIGDIAL*, pages 292–299.

Sungjin Lee and Maxine Eskenazi. 2013. Recipe for building robust spoken dialog state trackers: Dialog state tracking challenge system description. In *SIGDIAL*, pages 414–422.

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 *ACL*, pages 1437–1447.

Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. 2017. Learned in translation: Contextualized word vectors. In *NIPS*, pages 6294–6305.

Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural belief tracker: Data-driven dialogue state tracking. In *ACL*, pages 1777–1788.

Kevin P Murphy, Yair Weiss, and Michael I Jordan. 1999. Loopy belief propagation for approximate inference: An empirical study. In *UAI*, pages 467–475.

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *arXiv preprint arXiv:1812.00899*.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. 1999. The pagerank citation ranking: Bringing order to the web. Technical report, Stanford InfoLab.

Julien Perez and Fei Liu. 2017. Dialog state tracking, a machine reading approach using memory network. In *EACL*, pages 305–314.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *NAACL-HLT*, pages 2227–2237.

Osman Ramadan, Paweł Budzianowski, and Milica Gasic. 2018. Large-scale multi-domain belief tracking with knowledge sharing. In *ACL*, pages 432–437.

Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *ASRU Workshop*, pages 561–568.

Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards universal dialogue state tracking. In *EMNLP*, pages 2780–2786.

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2016. Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014a. A generalized rule based tracker for dialogue state tracking. In *SLT Workshop*, pages 330–335.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014b. The sjtu system for dialog state tracking challenge 2. In *SIGDIAL*, pages 318–326.

Blaise Thomson and Steve Young. 2010. Bayesian update of dialogue state: A pomdp framework for spoken dialogue systems. *Computer Speech & Language*, 24(4):562–588.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*, pages 5998–6008.

Zhuoran Wang and Oliver Lemon. 2013. A simple and generic belief tracking mechanism for the dialog state tracking challenge: On the believability of observed information. In *SIGDIAL*, pages 423–432.

TH Wen, D Vandyke, N Mrkšíc, M Gašíc, LM Rojas-Barahona, PH Su, S Ultes, and S Young. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449.

Jason D Williams. 2014. Web-style ranking and slu combination for dialog state tracking. In *SIGDIAL*, pages 282–291.

Jason D Williams and Steve Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Computer Speech & Language*, 21(2):393–422.

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 *ACL*, pages 808–819.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Kaige Xie, Cheng Chang, Liliang Ren, Lu Chen, and Kai Yu. 2018. Cost-sensitive active learning for dialogue state tracking. In *SIGDIAL*, pages 209–213.

Wenhan Xiong, Thien Hoang, and William Yang Wang. 2017. Deeppath: A reinforcement learning method for knowledge graph reasoning. In *EMNLP*, pages 564–573.

Puyang Xu and Qi Hu. 2018. An end-to-end approach for handling unknown slot values in dialogue state tracking. In *ACL*, pages 1448–1457.

Wen Zhang, Yang Feng, Fandong Meng, Di You, and Qun Liu. 2019. Bridging the gap between training and inference for neural machine translation. *arXiv preprint arXiv:1906.02448*.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive encoder for dialogue state tracking. In *ACL*, pages 1458–1467.

Lukas Zilka and Filip Jurcicek. 2015. Incremental lstm-based dialog state tracker. In *ASRU Workshop*, pages 757–762.