Multi-domain Dialogue State Tracking with Recursive Inference

Lizi Liao, Tongyao Zhu, Le Hong Long, Tat-Seng Chua
National University of Singapore, Singapore
liaolizi.llz@gmail.com,{tongyao.zhu,lehonglong}@u.nus.edu,dcscts@nus.edu.sg

ABSTRACT

Multi-domain dialogue state tracking (DST) is a critical component for monitoring user goals during the course of an interaction. Existing approaches have relied on dialogue history indiscriminately or updated on the most recent turns incrementally. However, in spite of modeling it based on fixed ontology or open vocabulary, the former setting violates the interactive and progressing nature of dialogue, while the later easily gets affected by the error accumulation conundrum. Here, we propose a Recursive Inference mechanism (**ReInf**) to resolve DST in multi-domain scenarios that call for more robust and accurate tracking capability. Specifically, our agent reversely reviews the dialogue history until the agent has pinpointed sufficient turns confidently for slot value prediction. It also recursively factors in potential dependencies among domains and slots to further solve the co-reference and value sharing problems. The quantitative and qualitative experimental results on the MultiWOZ 2.1 corpus demonstrate that the proposed **ReInf** not only outperforms the state-of-the-art methods, but also achieves reasonable turn reference and interpretable slot co-reference.

CCS CONCEPTS

• **Computing methodologies** → **Intelligent agents**; **Artificial intelligence**.

ACM Reference Format:
Lizi Liao, Tongyao Zhu, Le Hong Long, Tat-Seng Chua. 2021. Multi-domain Dialogue State Tracking with Recursive Inference. In *Proceedings of the Web Conference 2021 (WWW '21), April 19–23, 2021, Ljubljana, Slovenia.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3442381.3450134

1 INTRODUCTION

Starting from spoken dialogue system, the dialogue state tracking component in early days is difficult because automatic speech recognition (ASR) and spoken language understanding (SLU) errors are common. It thus tends to work on single domain setting with small fixed ontology [29]. Later, with great advancements in ASR and the availability of textual corpuses, many single-domain DST algorithms have been proposed [16, 21, 35] on textual inputs. However, single domain models are hard to scale to multi-domain setting which is more realistic in tracking user goals. When dialogue agents need to handle multiple tasks across different domains, the tracking problem becomes more complicated. It requires DST models to be more accurate in finding evidence for slot value prediction and more robust in handling the correlation among various slots.

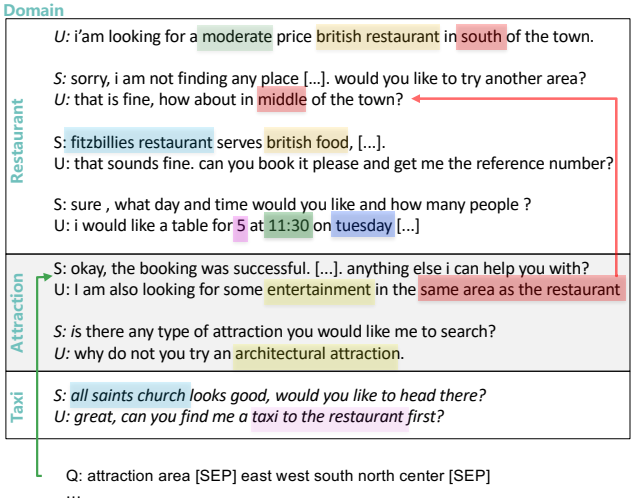

Figure 1: An example dialogue in MultiWOZ 2.1. Evidence appears in certain turns for specific domain slots. We also observe co-reference and value sharing among slots (in red).

Recent multi-domain DST methods try to address dialogue modeling in two different schemes: 1) Dialogue history scheme: taking the whole or window-sized dialogue history as input, it predicts slot value without explicitly discriminating over turns of utterances; 2) Turn by turn scheme: relying on the dialogue state generated for the previous turn and the most recent turn utterance(s), it learns to regenerate new state or partially update the previous state. However, in spite of different prediction mechanisms such as ontology-based or open-vocabulary based, both schemes are sub-optimal. The former dialogue history scheme usually treats dialogue history as a long sequence indiscriminately, while the conversation is naturally turn by turn progressively to reach an information alignment between user and agent. The conversation topic often jumps from one domain to another (73.8% in data, *e.g.*, from *restaurant* to *hotel* as shown in Figure 1), and the user requirements may also shift (7.8% in data, *e.g.*, from *cheap* to *moderate price*). As human rarely recall all their previous utterances but only reversely review the domain slot related turns, we expect our DST tracker to *selectively* review the dialogue history like us human during the conversation. The later turn by turn scheme actually works towards our *selective* design in the sense that it focuses on turn level. However, in this scheme, the error appeared in previous turn state will accumulate and largely affect later turns' prediction accuracy.

To accurately select the exact turns for a specific domain slot is a non-trivial task. For example, as illustrated in Figure 1, the agent needs to first resolve the domain for turns and find the turn containing values for the slot. More difficultly, there are nuanced

WWW '21, April 19–23, 2021, Ljubljana, Slovenia

ACM ISBN 978-1-4503-8312-7/21/04.
https://doi.org/10.1145/3442381.3450134

phenomena often occurred in natural conversations such as co-reference and value-sharing, *e.g.* "the same area as the restaurant" shown in Figure 1 refers to the value "center" for *restaurant-area*. The exact value "center" does not appear in the corresponding turn for *hotel-area* but refers to a previous correlated turn. Therefore, the agent needs to further infer the existence of co-reference and recursively obtain the corresponding shared value.

In this work, we formulate dialogue state tracking in multi-domain setting as Recursive Inference (**ReInf**) mechanism. As shown in Figure 1, the agent first forms queries for specific domain slots in state. Specifically, we construct two types of queries: multiple choice queries with limited number of value options and open-vocabulary queries which have a large or infinite number of value options. With the query, it infers whether it can ground on the most recent turn for value generation. If true, it will detect the value sharing slots and recursively infer the shared value to help the final value generation; If not, the agent will reversely review the former turns in dialogue history. The reverse review termination is that the agent detects value sharing which calls for query update, or it has backtracked to the beginning of dialogue history. With the turns found, we implement two versions of **ReInf**. One implementation can be end-to-end trained based on copy mechanism over BERT outputs. The other implementation consists of separate components where the generation module takes advantage of the pre-training model GPT-2 for value generation. Extensive experiments on the public large-scale dataset MultiWOZ 2.1 demonstrate the effectiveness of our proposed method. Our main contributions are concluded as follows.

- We propose a novel Recursive Inference (**ReInf**) strategy for multi-domain DST in task-oriented agents, pointing out its longing for accurate evidence finding.
- Instead of assuming independence among slots as previous works, we capture the potential dependencies across slots to handle the co-reference and value-sharing problem.
- We carry out extensive experiments and achieve state-of-the-art performance compared to other methods. The qualitative results also indicate that our tracker obtains reliable turn and slot reference during the recursive process.

2 RELATED WORK

2.1 Ontology-based v.s. Open-vocabulary based

Early DST models operate on a fixed ontology and perform prediction over a pre-defined set of slot-value pairs. They rely on hand-crafted features [26, 27], use convolutional neural networks [16], and try to enhance scalability [17–19, 34, 35]. Although performed well, these methods are hard to apply to more complex datasets or scenario such as multi-domain setting. It is often difficult to obtain a complete ontology for a task or domain. The idea of fixed ontology is also not sustainable as in real world applications they are subject to constant change. Therefore, recent approaches tend to be open-vocabulary based [4, 12, 31]. In general, we can organize them into span-based and generation-based methods. In span-based methods, they often treat DST as a machine reading comprehension problem. With dialogue history being the context and domain slot as the query, it extracts text span in dialogue history as the answer [4, 32, 36]. For instance, [32] applies an attention-based RNN with a pointer mechanism to extract values from the context. Such approach has its limitations. As mentioned in [33], there are many expressible values not found verbatim in the input, but rather mentioned implicitly, or expressed by a variety of rephrasings. Thus, they propose a hybrid mechanism to extract value or classify over candidates at the same time. [6] further predict values for slots by making use of three copy strategies. It combines the advantages of span-based methods and memory methods to avoid the use of value picklists.

An alternative to span-based methods is the value generation-based methods. [10, 31] propose to use an encoder-decoder architecture with copy mechanism to generate dialogue state values, which combines the distributions over a predefined vocabulary and the vocabulary of current dialogue history. [9] applies a similar mechanism for value generation, but takes the previous turn's dialogue state and the current turn utterances as inputs. Relying on the good performance of pre-trained, open domain language models such as GPT-2, [8] uses a single causal language model to train as a single sequence prediction problem, and has achieved the state-of-the-art performance. In this work, we also take advantage of the powerful GPT-2 in generation setting. However, instead of blindly relying on the whole dialogue history, we aim to accurately find the related turns containing evidence and leverage the slot correlations to generate the values.

2.2 History-based v.s. Turn-by-Turn based

Depending on the inputs to models, we can organize existing methods to history-based and turn-by-turn based. Most of current works apply the former scheme [4, 5, 11, 31]. They take the whole or window-sized dialogue history as input to recurrent neural networks or transformer models. For example, HJST considers the full dialogue history using a hierarchical recurrent neural network [5, 25]. More efforts concatenate different turns of dialogue history into a long sequence while use recurrent neural networks such as Bi-LSTM or RNN to encode it such as [4, 31]. There are also works inputting the whole history into BERT such as [33]. However, treating the whole dialogue history as a long sequence ignores the interactive nature of conversation. It also becomes hard to capture the user requirement updates during the conversation and the correlations among slots.

In order to capture the interactive nature of dialogue and improve the computational efficiency, there is another line of work in turn-by-turn style. Generally speaking, such methods take the previous turn's belief state and the current turn utterances as input to generate new dialogue state [1, 20]. For example, [1] leverages BERT model to extract slot values for each turn, then employs a rule-based update mechanism to track dialogue states across turns. [20] encodes previous dialogue state and current turn utterances using Bi-LSTM, then hierarchically decodes domains, slots and values one after another. At the same time, [9] encodes these inputs with BERT model while predicts operation gates and generates possible values for each slot. Similarly, [13] proposes to incrementally infer new dialogue state from previous dialogue state and newly extracted turn results via BERT with the help of database knowledge. However, all these methods naturally suffer from the error accumulation problem as the errors in the previous dialogue state will largely affect the later turns' results.

In this work, we argue that both whole history-based and turn-by-turn based methods are not good enough. Since conversations are changing in domain and user requirements frequently, it would be more natural to *selectively* review the dialogue history like human during the conversation. Moreover, most of current works treats different domain slots relatively independently. However, we observe that fluent slot correlations such as slot co-references can benefit model performance.

3 METHOD

In this section, we formally introduce the multi-domain dialogue state tracking task and our proposed Recursive Inference (ReInf) approach. The task of multi-domain dialogue state tracking is defined as follows. Formally, we represent a dialogue X as $X = \{U_1^a, U_1^u, \cdots, U_T^a, U_T^u\}$, where U_t^a is the agent utterance in turn t and U_t^u is the user utterance in turn t. Each turn t is associated with a dialogue state Y_t which is the target that our model needs to predict. It is a set of (domain, slot, value) tuples. Each tuple in Y_t represents that, up to the current turn t, a slot s of domain d, which takes the value v has been provided in the interaction. In the multi-domain DST, there are several domains. Each domain d has its slots, and each slot s has its possible value candidates V^s. For example, the *hotel* domain has a slot named *price range* which can take possible values like `moderate, cheap`, and `expensive`. Some slots do not have pre-defined values, *i.e.*, V^s is missing in the domain ontology. For example, the *taxi* and *train* domains have slots such as *arrive time* and *leave time* which are hard to enumerate all possible values. Following the convention of MultiWOZ 2.0 and MultiWOZ 2.1, we use the term "slot" $-$ s to refer to the concatenation of a domain name and a slot name, such as *hotel-price range*.

Next, we first provide an overview of ReInf in Section3.1, followed by Section 3.2 introducing its MATCH, SHARE and GEN modules. Different versions of ReInf are given in Section 3.2.

3.1 Recursive Inference

Algorithm 1: Recursive Inference (**ReInf**)

Function `ReInf`$(X_t, s, t, evidences)$:
- **if** $t < 1$ **then**
  - **return** $GEN(evidences, s)$ // reversely reviewed all turns
- $confi \leftarrow MATCH(U_t^a, U_t^u, s)$ // test the current turn
- **if** $confi$ **then**
  - /* feel confident at this turn */
  - $SS \leftarrow SHARE(U_t^a, U_t^u, s)$ // get shared slots
  - **foreach** $s' \in SS$ **do**
    - $VV \leftarrow$ `ReInf`$(X_{t-1}, s', t-1, \phi)$ // get value recursively
  - $evidences \leftarrow ([U_t^a, U_t^u], SS, VV)$ // add to evidences set
  - /* stop reverse if sharing slot is detected or have obtained enough evidences */
  - **if** $s' \neq null$ *or* $|evidences| > threshold$ **then**
    - **return** $GEN(evidences, s)$
- /* recursively review former turns */
- $t' \leftarrow t - 1$
- **return** `ReInf`$(X_{t'}, s, t', evidences)$

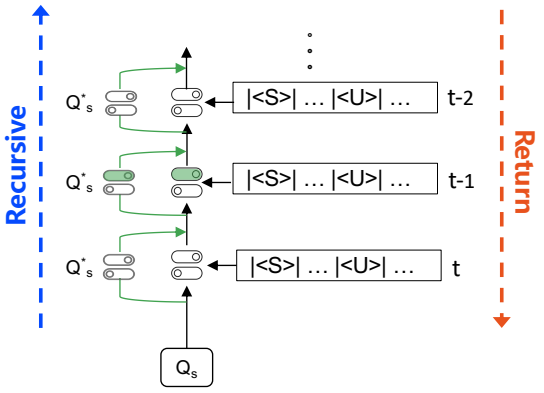

Figure 2: An abstract view of Recursive Inference mechanism. The down-to-up direction (bashed blue) represents recursive call over turns (consists of system utterance |<S>| and user utterance |<U>|). It not only includes reverse review of turns, but also captures recursive inference of value for shared slots Q_s^*.

First of all, the overall structure of the proposed Recursive Inference mechanism (ReInf) is shown in Algorithm 1. As illustrated, t is the present turn for which the model needs to predict dialogue state and X_t denotes the dialogue history till the t-th turn. Here s is a specific slot our tracker needs to predict value for. Given any slot s, our state tracker will first construct a question Q_s and test whether it *matches* with the most recent turn utterances for evidence finding (more details will be given in Section 3.2). If not, our tracker will reversely review former turns. This process will be kept executing until the tracker detects value sharing, has obtained enough evidences, or has backtracked to the beginning of dialogue. If our tracker finds a confident turn t_p, it will first find the slots that might share value with s, and recursively find the value for each shared slot s' based on the new questions $Q_{s'}$ and the subset of dialogue history before this turn. It then *generates* the value for slot s based on the turn utterances found and shared values.

In addition, we give a high level illustration of Recursive Inference mechanism (ReInf) in Figure 2. Intuitively, all the switches on both the trunk and branches are initially open (*i.e.*, turned off). Our ReInf is recursively called from the present to the past, closing the switch on the trunk one by one until the recursion terminates. The switches on the trunk decide whether to further review previous turns. At the same time, our ReInf also closes the switch on branches when value sharing is detected. These switches on branches decide whether there exist shared slots that need to further predict value for these slots recursively. When the recursion termination condition is met, we unroll the process from the past to present and finally predict the value for the current slot.

In order to achieve the recursive inference algorithm, we further design three modules, *i.e.*, MATCH, SHARE and GEN. Generally speaking, MATCH module asserts the recursion termination condition and pinpoints the evidence turns for value generation. SHARE module infers the slots that might share value with the current slot s based on the turn utterances found. GEN module generates the value for specific domain slots and considers various language phenomena such as co-reference and value sharing.

3.2 Neural Modules

Algorithm 2: MATCH Module

Function MATCH(U_t^a, U_t^u, s):
- $Q_s \leftarrow construct(s)$
- $\mathbf{q}_s \leftarrow f_{BERT}^{CLS}(Q_s)$
- $\mathbf{H}_U \leftarrow f_{BERT}([U_t^a, U_t^u])$
- $\boldsymbol{\alpha} \leftarrow Softmax(\mathbf{H}_U \cdot (\mathbf{q}_s)^T)$ // the query attends to utterances
- $confi \leftarrow Softmax(\mathbf{W}_{match} \cdot (\boldsymbol{\alpha} \cdot \mathbf{H}_U)^T)$
- **return** confi

3.2.1 MATCH Module. The MATCH module is designed to determine whether the current turn contains evidence for value. Specifically, we treat dialogue state tracking as a question answering problem which can be solved by machine reading methods. Thus, MATCH module treats the slot s as query and predicts whether a turn of utterances U_t^a, U_t^u contains evidence of appropriate value.

Inspired from [36], $construct(s)$ generates question Q_s for the slot s. For each slot s where there exists a predefined value set V^s, we construct a question $Q_s = \{s, V^s, don't\ care, not\ mentioned\}$. For example, the constructed question for the slot 'hotel-price range' will be $Q_s = \{hotel, price\ range, cheap, moderate, expensive, don't\ care, not\ mentioned\}$. It represents the following natural language question: "Is the 'price range' of the 'hotel' mentioned? If so, which of the following option is correct: A) cheap, B) moderate, C) expensive, D) don't care." In the case that V^s is not available such as for slots like 'hotel-name' or 'taxi-arrive time', we construct the question $Q_s = \{s, don't\ care, not\ mentioned\}$. For example, the constructed question for the slot 'taxi-arrive time' will be $Q_s = \{taxi, arrive\ time, don't\ care, not\ mentioned\}$. It represents the natural language question "Is the 'arrive time' of 'taxi' mentioned? If so, what is the 'arrive time' preferred?". With the constructed question Q_s, we obtain its vector representation $\mathbf{q}_s$ via the [CLS] position's output from a BERT encoder. Meanwhile, we also get the contextualized representations of words $\mathbf{H}_U = [\mathbf{h}_1, \mathbf{h}_2, \cdots, \mathbf{h}_L]$ via BERT encoder using concatenated turn utterances $[U_t^a, U_t^u]$ with length L of words.

$$\boldsymbol{\alpha} = Softmax(\mathbf{H}_U \cdot (\mathbf{q}_s)^T). \quad (1)$$

$$\mathbf{confi} = Softmax(\mathbf{W}_{match} \cdot (\boldsymbol{\alpha} \cdot \mathbf{H}_U)^T). \quad (2)$$

In the MATCH module, it first takes the question feature $\mathbf{q}_s$ to attend over the utterance words $\mathbf{H}_U$ as in Equation 1. Then, it weighted sums together the evidence from utterance words and predicts whether the turn contains value information for the question or not as in Equation 2. The result $\mathbf{confi} \in \mathbb{R}^2$ indicates the certainty and uncertainty of the turn containing evidence. We regard the value in first dimension as the confidence score. The $\mathbf{W}_{match}$ is a weight matrix for fully connected layer. For ease of illustration, the bias terms are ignored.

3.2.2 SHARE Module. The SHARE module detects the shared slots for slot s based on the turn utterances U_t^a, U_t^u. The $candidates(s)$ function provides candidate sharing slots. For example, given the s as 'hotel-area', it will return candidates 'restaurant-area' and 'attraction area'. We obtain this function via statistics in training data.

Algorithm 3: SHARE Module

Function SHARE(U_t^a, U_t^u, s):
- $S' \leftarrow candidates(s)$
- **foreach** $s' \in S'$ **do**
  - $\mathbf{q}_{s'} \leftarrow f_{BERT}^{CLS}(Q_{s'})$
  - $\mathbf{H}_U \leftarrow f_{BERT}([U_t^a, U_t^u])$
  - $\boldsymbol{\beta} \leftarrow Softmax(\mathbf{H}_U \cdot (\mathbf{q}_{s'})^T)$ // the query attends to utterances
  - $share \leftarrow Softmax(\mathbf{W}_{share} \cdot (\boldsymbol{\beta} \cdot \mathbf{H}_U)^T)$
  - **if** *share* **then**
    - $SS.insert(s')$
- **return** SS

Basically, we harvest all the slot pairs that have shared value in the same dialogue. Then, for each candidate slot s', we generate its question representation $\mathbf{q}_{s'}$ and attend to the utterance representations $\mathbf{H}_U$. The $\boldsymbol{\beta}$ refers to the attention weights. The attended representations will go through a fully connected layer with parameter $\mathbf{W}_{share}$ and predict whether s' shares value with s or not (as in Equation 3). For example in Figure 1, for the candidate slot 'restaurant-area', the turn utterances $[U_5^a, U_5^u]$ contain the phrase *same area as the restaurant*. Its attention weight will be large and help to predict 'restaurant-area' as a shared slot.

$$\mathbf{share} = Softmax(\mathbf{W}_{share} \cdot (\boldsymbol{\beta} \cdot \mathbf{H}_U)^T). \quad (3)$$

The result $\mathbf{share} \in \mathbb{R}^2$ indicates whether s' shares value with s or not. We regard the value in first dimension as the probability score of sharing. In Algorithm 3, SS refers to the set of all slots that share value with s.

Algorithm 4: GEN Module

Function GEN($evidences, s$):
- $\mathbf{q}_s \leftarrow f_{BERT}^{CLS}(Q_s)$
- $\mathbf{H}_U \leftarrow f_{BERT}(evidences.U)$ // concat utters in evidences
- **foreach** $(s', v') \in (evidences.SS, evidences.VV)$ **do**
  - $Str \leftarrow [Str, s', v']$
- $\mathbf{H}_{Str} \leftarrow f_{BERT}(Str)$
- $\mathbf{H} \leftarrow [\mathbf{H}_U, \mathbf{H}_{Str}]$ // concat uttrs and shared slot value string matrix
- /* generate the value word by word using copy mechanism */
- **while** $\mathbf{e}_k$ *does not refer [EOS] token* **do**
  - $\mathbf{g}_k \leftarrow GRU(\mathbf{g}_{k-1}, \mathbf{e}_k)$
  - $\mathbf{p}_k^{vocab} \leftarrow Softmax(\mathbf{E} \cdot (\mathbf{g}_k)^T)$
  - $\mathbf{p}_k^{copy} \leftarrow Softmax(\mathbf{H} \cdot (\mathbf{g}_k)^T)$
  - $\mathbf{c}_k \leftarrow \mathbf{p}_k^{copy} \cdot \mathbf{H}$
  - $\boldsymbol{\gamma} \leftarrow Sigmoid(\mathbf{W}_c \cdot [\mathbf{g}_k, \mathbf{e}_k, \mathbf{c}_k])$
  - $\mathbf{p}_k^{gen} \leftarrow \boldsymbol{\gamma} \cdot \mathbf{p}_k^{vocab} + (1 - \boldsymbol{\gamma}) \cdot \mathbf{p}_k^{copy}$
- **return**

3.2.3 GEN Module. The GEN module is designed to predict the value for slot s given the found turn utterances and shared values. We apply the soft gated copy mechanism [23] to get the final output distribution $\mathbf{p}_{s,k}^{gen}$ over the candidate value tokens for s. We generate

the value word by word until the [EOS] token in the way similar to [9]. Specifically, we use Gated Recurrent Unit (GRU) [2] decoder like [31]. The GRU recurrently updates the hidden state $\mathbf{g}_{s,k}$ by taking a word embedding $\mathbf{e}_{s,k}$ as the input until the [EOS] token is predicted:

$$\mathbf{g}_{s,k} = GRU(\mathbf{g}_{s,k-1}, \mathbf{e}_{s,k}). \tag{4}$$

It is initialized with $\mathbf{g}_{s,0} = tanh(\mathbf{W}_{pool} \cdot (\mathbf{h}_U)^T)$ and $\mathbf{e}_{s,0} = \mathbf{q}_s$, where $\mathbf{h}_U$ is the [CLS] position's output vector from the BERT model with concatenated utterances as input. $\mathbf{g}_{s,0}$ can be seen as the aggregated sequence representation of the utterances obtained from a feed-forward layer with learnable weights $\mathbf{W}_{pool}$.

As the decoder works on, its hidden state is first transformed to the probability distribution over the vocabulary as in Equation 5, where $\mathbf{E}$ is the word embedding matrix shared across the encoder and the decoder,

$$\mathbf{p}^{vocab}_{s,k} = Softmax(\mathbf{E} \cdot (\mathbf{g}_{s,k})^T). \tag{5}$$

At the same time, the hidden state is used to compute the history attention $\mathbf{p}^{copy}_{s,k}$ as in Equation 6 over the encoded sequence $\mathbf{H}$ that contains utterances and shared slot values:

$$\mathbf{p}^{copy}_{s,k} = Softmax(\mathbf{H} \cdot (\mathbf{g}_{s,k})^T). \tag{6}$$

The final output distribution $\mathbf{p}^{gen}_{s,k}$ is the weighted-sum of two distributions

$$\mathbf{p}^{gen}_{s,k} = \boldsymbol{\gamma} \cdot \mathbf{p}^{vocab}_{s,k} + (1 - \boldsymbol{\gamma}) \cdot \mathbf{p}^{copy}_{s,k},$$
$$\boldsymbol{\gamma} = Sigmoid(\mathbf{W}_c \cdot [\mathbf{g}_{s,k}, \mathbf{e}_{s,k}, \mathbf{c}_{s,k}]),$$

where $\mathbf{W}_c$ is a learnable weight matrix and $\mathbf{c}_{s,k} = \mathbf{p}^{copy}_{s,k} \cdot \mathbf{H}$ is the context vector.

3.3 Two Implementation Versions of ReInf

The core of Recursive Inference is to reversely review turns for finding confident evidences and recursively obtain shared slot values for final value generation. With the copy-based GEN module illustrated in Section 3.2, ReInf can be implemented and trained end-to-end. At the same time, motivated by the strong generation performance of GPT-2, we implement a new version of ReInf by realizing GEN with pre-trained GPT-2 model. Thus, this version consists of separate components: MATCH and SHARE predict turn and shared slots for GEN, while GEN focuses on value generation.

3.3.1 ReInf in End-to-End Style. In the end-to-end implementation of ReInf, we train the MATCH, SHARE and GEN modules separately first and fine-tune them together for each training instance (X_t, Y_t). We show the objective for one slot s here. We denote t_s as the ground truth turn index, SS as the shared slots ground truth and y_s as the ground truth value. We use $\mathbf{o}_i$ as the two dimensional one-hot vector for turn ground truth for the i-th turn. Then $\mathbf{o}_{t_s}$ will be [1,0] while others being [0,1]. The objective for MATCH is as in Equation 7. For the j-th shared slot in SS, we construct one-hot vector as $\mathbf{v}_j$, thus the objective for SHARE is the average as in Equation 8. We use K_s as the number of tokens in y_s and $\mathbf{y}_{s,k}$ as the one-hot vector for the ground truth token at the k-th decoding step. The objective function for GEN is the average of the negative log-likelihood in Equation 9.

$$L_{MATCH} = -\frac{1}{t - t_s + 1} \sum_{i=t_s}^{t} log(\mathbf{confi}_i \cdot (\mathbf{o}_i)^T), \tag{7}$$

$$L_{SHARE} = -\frac{1}{|SS|} \sum_{j=SS_i} log(\mathbf{share}_j \cdot (\mathbf{v}_j)^T), \tag{8}$$

$$L_{GEN} = -\frac{1}{S} \sum_{s \in S} [\frac{1}{K_s} \sum_{k=1}^{K_s} (\mathbf{y}_{s,k})^T log(\mathbf{p}^{gen}_{s,k})]. \tag{9}$$

Therefore, the final joint loss L_{joint} for the slot s of the sample to be minimized is the sum of the losses mentioned above:

$$L_{joint} = L_{MATCH} + L_{SHARE} + L_{GEN}. \tag{10}$$

3.3.2 ReInf with Separate Components. In the realization of ReInf with separate components, we first train the MATCH and SHARE modules together as in Equation 10 but without L_{GEN}. We then train the GPT-2 based GEN module separately. We form each single training sequence as $z = [U^a_{t_s}, U^u_{t_s}, SS_1, V_1, \cdots, SS_{|SS|}, V_{|SS|}, s, value]$, where *value* is the target the GEN aims to generate. We also add specific separator tokens among different types of elements in z. Such formalization allows us to model the joint probability over the sequence z, *i.e.*, constructing a language modeling goal to learn $p(z)$. By factorizing this distribution using the chain rule of probability $p(z) = \prod_{i=1}^{|z|} p(z_i|z_{<i})$, we fine-tune GPT-2 network with parameters θ to minimize the negative log-likelihood over a set Z of such samples,

$$L_{GEN} = -\sum_{z \in Z} \sum_{i=1}^{|z|} log\ p_\theta(z_i|z_{<i}).$$

3.3.3 Filter for Improving Efficiency. As in belief state Y_t, 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 [31] to filter out such slots first, for which we can skip the ReInf process and predict the value *not mentioned* directly. We apply the separate training objective as the cross entropy loss computed between the predicted slot gate $\mathbf{p}^{gate}_s$ and the true one-hot label $\mathbf{q}^{gate}_s$ as below:

$$L_{gate} = -log(\mathbf{p}^{gate}_s \cdot (\mathbf{q}^{gate}_s)^T),$$

where for gate prediction, we calculate $\mathbf{H}_{X_t} = f_{BERT}(X_t)$ as contextualized word representations for dialogue history, and then apply query attention to classify whether the slot should be filtered,

$$\boldsymbol{\eta} = Softmax(\mathbf{H}_{X_t} \cdot (\mathbf{q}_s)^T),$$
$$\mathbf{p}^{gate}_s = Softmax(\mathbf{W}_{gate} \cdot (\boldsymbol{\eta} \cdot \mathbf{H}_{X_t})^T).$$

In addition, we also add an auxiliary task to force the filter to learn the correlation between slots and dialogue domain, by applying the cross entropy loss between the predicted domain $\mathbf{d}_{X_t}$ of dialogue history and the true label vector $\hat{\mathbf{d}}_{X_t}$,

$$L_{dom} = -log(\mathbf{d}_{X_t} \cdot (\hat{\mathbf{d}}_{X_t})^T),$$

where the domain classification is done with a softmax layer on top of $\mathbf{h}_{X_t} = f^{CLS}_{BERT}(X_t)$ which takes the [CLS] position output vector from BERT,

$$\mathbf{d}_{X_t} = Softmax(\mathbf{W}_{dom} \cdot (\mathbf{h}_{X_t})^T).$$

4 EXPERIMENTS

4.1 Datasets and Setup

We carry out experiments on MultiWOZ 2.1 [3], 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 7 domains: *attraction, restaurant, hotel, train, taxi, hospital* and *police*. 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. In total, there are 30 domain-slot pairs and over 4,500 possible values, which is different from existing standard datasets like WOZ [28] and DSTC2 [7] – with less than 10 slots and only a few hundred values.

4.2 Training Details

We adopt the pretrained bert-base-uncased version of BERT and initialize the learning rate for fine-tuning as $3e$-5. The hidden size of the decoder is the same as that of the encoder, which is 768. We use BertAdam as our optimizer and use greedy decoding for slot value generator. For GPT-2, the input to the model is tokenized with pre-trained BPE codes [24] associated with DistilGPT2 [22], a distilled version of GPT-2. Similar to SimpleTOD, we use default hyperparameters for GPT-2 and DistilGPT2 in Huggingface Transformers [30]. Sequences longer than 1024 tokens are truncated.

4.3 Evaluation Metrics

Similar to [31], we adopt the joint accuracy to evaluate the performance on multi-domain DST. The joint accuracy compares the predicted belief states to the ground truth at each turn t. The prediction is considered correct if and only if all values exactly match the ground truth values. Besides it, we also report the turn prediction accuracy of our method. It evaluates the ability of our model in finding turns that contain useful evidence. For each slot, we also provide its overall error rate to show detailed comparison.

4.4 Comparing Models

We denote the two versions of ReInf implementation as ReInf$_{EE}$ and ReInf$_{SC}$, which corresponds to ReInf in end-to-end style and ReInf with separate components respectively. They are compared with the following models: DSTRead [4], TRADE [31], SOM [9], COMER [20], DS-DST [33], TripPy [6] and SimpleTOD [8]. More details about these methods are given below:

- **DSTRead** [4]: It treats dialogue state tracking as a reading comprehension problem. Given the whole dialogue history, it learns to extract slot values in text spans. It exploits contextual word embeddings and explicitly tracks whether a slot value should be carried over to the next turn.
- **TRADE** [31]: 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.

Comparing Methods	Joint Accuracy
DSTRead	0.364
TRADE	0.456
SOM	0.530
COMER	0.362
DS-DST	0.533
TripPy	0.553
SimpleTOD	0.557
ReInf$_{EE}$	0.520
ReInf$_{SC}$	**0.583**

Table 1: The multi-domain DST evaluation on MultiWOZ 2.1 dataset. The proposed ReInf achieves the best performance.

- **SOM** [9]: It works in turn-by-turn style by considering dialogue state as an explicit fixed-sized memory, and adopts a selectively overwriting mechanism for generating values with copy mechanism.
- **COMER** [20]: It also takes current turn utterances with previous turn belief state as input. A hierarchical encoder-decoder structure is applied to generate a sequence of belief states more efficiently.
- **DS-DST** [33]: 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 [4].
- **TripPy** [6]: Relying on the whole dialogue history as input, it makes use of various copy mechanisms to fill slots with values. Value sharing is considered over the dialogue history.
- **SimpleTOD** [8]: This is the current state-of-the-art model on the multi-domain MultiWOZ 2.1 dataset. Based on the whole dialogue history, it uses a single pre-trained GPT-2 model to fine-tune on all sub-tasks as a single sequence prediction problem. We use the DST part for comparison.

4.5 Quantitative Results

We first compare our model with a wide range of 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, ReInf$_{SC}$ improves the performance by 10.0% as compared to the state-of-the-art model SOM in turn-by-turn style. It also improves the performance by 4.67% as compared to the state-of-the-art model SimpleTOD in history-based style. It validates our call for accurate evidence finding.

When the whole dialogue history is fed to the models, it tends to lose focus especially when user requirements changes or slots with similar value candidates appeared. For methods like DSTRead and TRADE, RNN models are leveraged to encode the dialogue history. However, RNN models suffers from the well-known gradient vanishing problem. Since domain in dialogue contents frequently change, information appeared in former turns might be easily get overwhelmed by later turns. When generating the dialogue summary, value for such slots might get misled easily. Indeed, Transformer-based models are better in learning contextualized representations, thus more recent works such as DS-DST, TripPy

Figure 3: Turn prediction accuracy of ReInf$_{_SC}$ for each slot that is predicted to have a value.

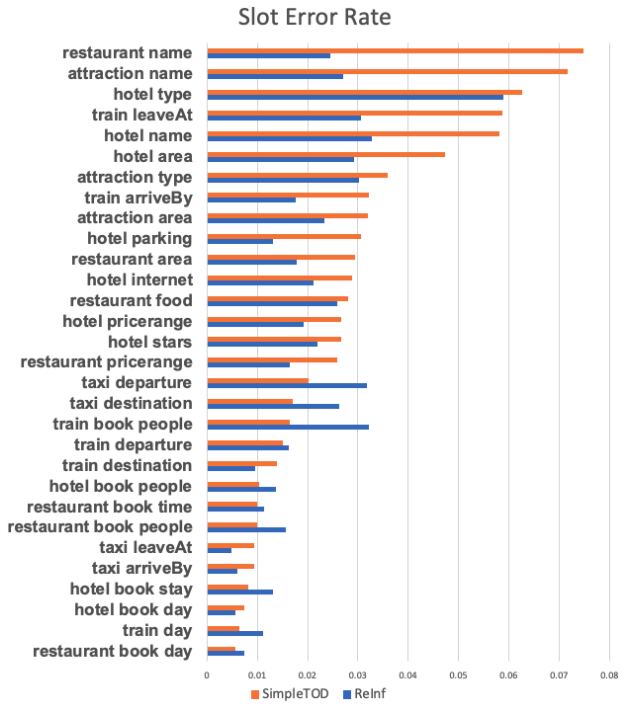

Figure 4: Slot error rates on the test set for SimpleTOD and the proposed ReInf (we report the rates for ReInf$_{_SC}$ here).

and SimpleTOD are all based on Transformer models to encode the whole dialogue history. For example, the DS-DST directly takes in slot augmented dialogue history while SimpleTOD simply takes in the long sequence of dialogue history. Although the great strength of Transformer models contribute to the better performance of these methods, blindly working on the whole history indiscriminately would introduce noise as well as confuse the model, especially in multi-domain setting.

When working on the most updated turn utterances with previous turn dialogue states, the model will be largely affected by error accumulation. For instance, the COMER hierarchically generates domain, slot and value one after another by taking former turn dialogue state as part of the input. When there is any error appeared in the state, it will be carried to the current state generation. The SOM method designs a gate to further decide whether to carryover, delete or update certain slot-value pairs from the former turn. However, it still cannot avoid the error accumulation conundrum, thus resulting in lower performance than ReInf.

Note that all the comparing methods do not model the relation among slots explicitly except the TripPy. It is the first one to explicitly consider the value sharing among slots. However, it takes the whole dialogue history as input to predict which slot shares value with a specific one. There will be much noise to overwhelm the sharing slot prediction. In our method, we focus on specific turns to do slot sharing prediction. Also, we make use of statistic information as a general candidate slots generator which further helps to reduce noise. We will show later in Subsection 4.6.3 that the sharing slots obtained are reasonable.

Among all these methods, SimpleTOD is a rather simple and straight-forward design. However, it achieves the best performance

Settings	Joint Accuracy
all history	0.557
selective turns	0.561
selective turns + recursive share	**0.583**

Table 2: Ablation study results on selecting turns and recursively inferring values from shared slots.

in baselines. This attributes to the strong capability of pre-trained GPT-2. We also notice that ReInf$_{_EE}$ can not beat ReInf$_{_SC}$. There might be two reasons: Firstly, ReInf$_{_SC}$ implementation takes advantage of the pre-trained GPT-2 model in the GEN module. Secondly, the encoder of ReInf$_{_EE}$ is based on pre-trained BERT model while the decoder is based on GRU, we suspect that such discrepancy would affect the performance. We report here as a negative example.

4.5.1 Ablation Study. To further investigate the ReInf mechanism, we carry out ablation study of ReInf$_{_SC}$ on the selective turns design and the recursive sharing value design as in Table 2. First of all, we report the results when using the whole dialogue history as input. In ReInf$_{_SC}$, as the GEN module is separately trained, when using the whole dialogue history, it actually degrades to the SimpleTOD model. When we just rely on the turns selected by MATCH while ignore the other turns, we observe a performance improvement for the setting *selective turns*. This is because many noise information got filtered out. The model will be more focused on the turns containing the value evidence. Furthermore, we append the shared slots and value to the input, we observe a further improvement on the *selective turns+recursive share* setting. This is as expected since sometimes the user will not mention the exact value for a specific slot such as *attraction-area*, but mention as *same*

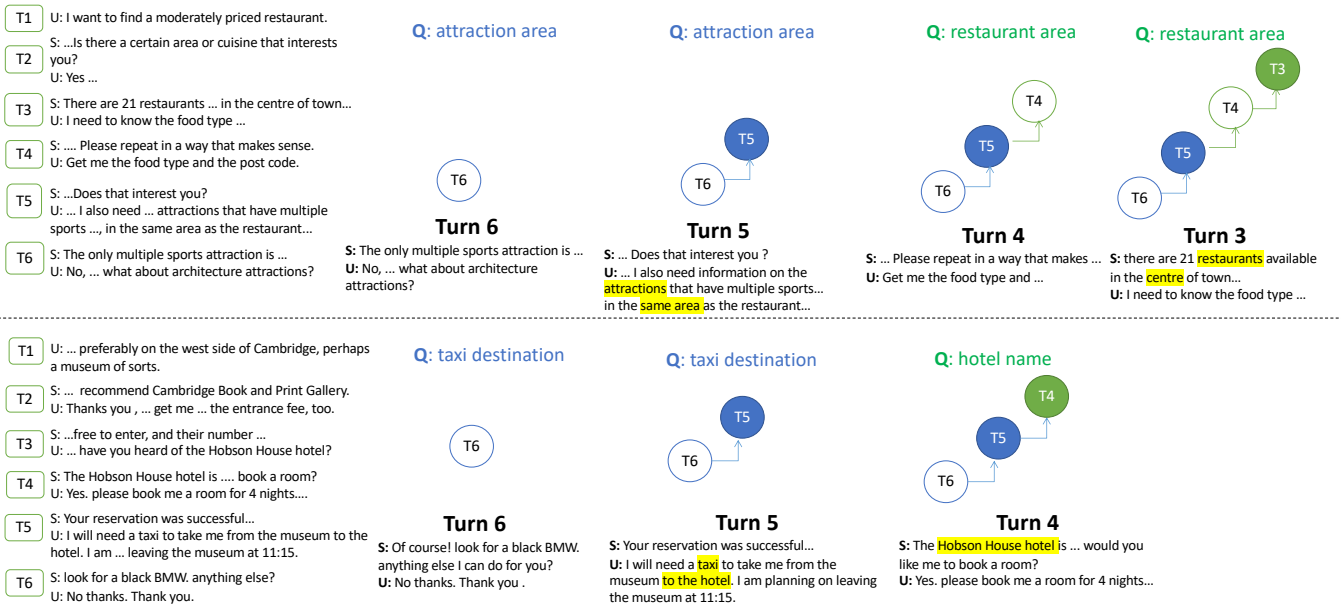

Figure 5: Case study on recursive inference. The filled circles are the turns found by ReInf for specific query slots. Our model successfully obtains the turns containing value evidence by reversely reviewing the dialogue history. It manages to detect the sharing value slots and recursively infer its values (the color of query circle changes when sharing slots are detected).

area as the restaurant. By appending the value of *restaurant-area* to the input, it actually provides the model with more evidence and helps GPT-2 to generate value that is not shown in utterances.

Figure 3 shows the accuracy of turn prediction of each slot. The turn prediction accuracy is based on instances where a slot is predicted to have a value. Generally speaking, ReInf obtains good performance on finding turns. However, we also notice that it gets lower accuracy for several slots related to hotel such as *hotel-type*. We suppose that this might be because the value of such slot often got mentioned several times in different turns. For instance, the user might first give information about hotel type in a turn, and later receive hotel name which also indicates the type being hotel.

Figure 4 shows slot error rate of ReInf$_{SC}$ and SimpleTOD on the test set for more detailed comparison. We observe that ReInf$_{SC}$ improves over SimpleTOD in most of the slots. The most obvious ones are the name slots, such as *restaurant-name*, *attraction-name* and *hotel-name*. We suspect that the main reason is that these slots get open-vocabulary values and the length of these values is relatively longer than other slots. Providing with dense selected turns instead of the whole dialogue history while also enriched with shared values, the GPT-2 model will be more focused and would learn to generate these better. We notice that slots like 'taxi destination' and 'taxi departure' get worse. The reason behind this is that they often appear in the same turn (85.6% of the times, see the second example in Figure 5). Therefore, in shared slots, both attraction name and hotel name will be found.

4.6 Qualitative Results

4.6.1 Turn Prediction Accuracy and Slot Error Analysis. In this Subsection, we first demonstrate the capability of the proposed ReInf in accurate turn finding and slots sharing via recursion cases. We then provide details of statistics comparisons in more general sense.

4.6.2 Recursion Case Study. In Figure 5, two examples are shown to demonstrate how the proposed ReInf finds the correct turns and sharing value slots of a given slot. In the first example on the top, the model needs to generate value for the slot *attraction-area* given six turns of utterances. Starting from the Turn 6, it reversely checks the turns until it gets confident that Turn 5 contains information about *attraction-area*. However, the user's utterance in Turn 5 refers to another slot for the exact value. Therefore, the query of interest is updated to *restaurant-area*, and then ReInf recursively traverses in history to find the Turn 3 eventually. The model's prediction of the new query *restaurant-area* will be used for the value generation of *attraction-area*, and our model is able to correctly predict 'center' in the this example. In the second example on the bottom, the model starts with query about *taxi-destination*. It finds Turn 5. However, in Turn 5 the user also refers to the hotel mentioned before. In order to find the *hotel-name*, the model steps back to history and finds the *hotel-name* in Turn 4, which will then be used for the prediction of *taxi-destination*. Through these two examples, the model's ability to recursively infer along dialogue history is clearly shown. It largely simulates human behavior when attempting to find a value of a slot. Note that in the reversely reviewing process and recursive inference process, the model will end its searching process as long as enough evidence turns are found or sharing slots are detected. It is not necessary to traverse all the turns in the dialogue history laboriously. For instance, it will stop in the Turn 3 for the first example and stop in the Turn 4 for the second example with evidence *threshold* being 1.

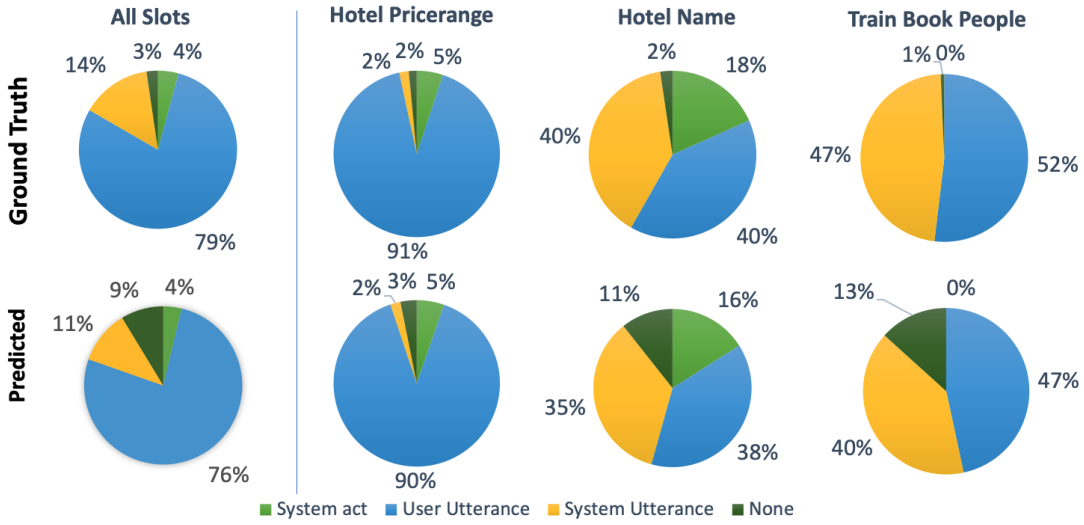

Figure 6: Comparison of general statistics of evidence position between predicted results and the ground truth. We search in the sequence of system act, system utterance, user utterance.

4.6.3 Comparison of General Statistics. One might argue that several special examples are not representative enough to support the model's overall validity. We thus provide some general statistics to compare the predicted results of ReInf$_{SC}$ and the ground truth. First of all, in order to show whether the proposed model generally finds the correct turns, we count the statistics of the value generated in the position of found turns. We then compare the statistics with the ones harvested from ground truth results. We examine the value location of a slot, and determines whether the value of a slot appears first in system acts, in system utterance, or in user utterance. In the test set, around 79% of ground-truth values of slots come from user utterance, which is reasonable as the user is often taking initiatives to provide information. We show the comparison over all slots and some example slots in Figure 6. As shown in the pie charts of 'all slots', for predictions of our model, the distribution of value locations are generally similar to that of the ground truth. Only the ratio for 'none' is relatively higher. This is reasonable as our model gets wrong turn prediction sometimes. Under such cases, we might not be able to find the value in the turn. In addition to statistics of 'all slots', there are also breakdowns according to individual slots with different patterns. For instance, the value of *hotel-price range* is mostly provided by the user, and our models' predictions follow a similar pattern. In contrast, the evidence for *hotel-name* can be retrieved from both system part or user part. The predictions by our model also follow a similar pattern that the evidence is more evenly distributed among different locations in a turn. Lastly, for the slot *train-book people*, both system and user utterance are equally likely to contain evidence. This might be because the system might repeat order details when asking for confirmation. We also observe similar patterns for this. Overall, our model is able to utilize information from different parts of a turn and generate values based on evidence found.

We further investigate the similarity between frequency of our predictions on value-sharing slots and the ground-truth statistics, as shown in Figure 7. If a slot shares the same value with another slot in a turn, then it would be considered a value-sharing instance.

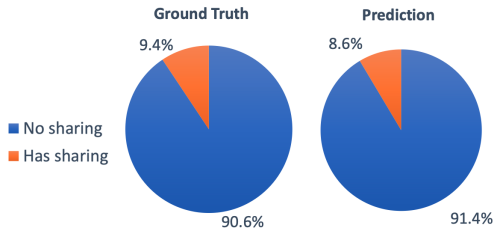

Figure 7: Comparison of general statistics of slots sharing between the ground truth and predicted results.

In our model's prediction, around 8.6% of the time a slot shares value with another slot in the same turn, which is comparable to the 9.4% of ground-truth frequency. We therefore expect that our model has generally learnt to detect the value sharing situation well and this provides a good base for it to copy value from shared slots.

5 CONCLUSION

We have demonstrated that our approach can handle challenging multi-domain DST task. We designed a Recursive Inference mechanism (ReInf) consisting of three straight-forward neural modules that determine the recursion at run-time. Our dialogue state tracker reversely reviews the slot-related history to find the turns containing evidence for value generation, and recursively infers values for slots that share value with the current slot. In order to investigate the various effect of current Transformer models on ReInf, we realize two implementations of ReInf: one with the copy mechanism for value generation based on BERT outputs in end-to-end style while the other simply based on the pre-tained GPT-2 via separate components. Experimental results on the large-scale multi-domain dataset MultiWOZ 2.1 demonstrate that our proposed model not only achieves state-of-the-art performance, but also obtains accurate turn findings, reasonable slot recursion and explainable value generation. Moving forward, we are going to investigate possible ways to get the turn finding and GPT-2 part trained together to facilitate downstream tasks such as response generation [14, 15].

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

[1] Guan-Lin Chao and Ian Lane. 2019. BERT-DST: Scalable End-to-End Dialogue State Tracking with Bidirectional Encoder Representations from Transformer. *Proceedings of the 20th Annual Conference of the International Speech Communication Association*, 1468–1472.

[2] Kyunghyun Cho, Bart van Merriënboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014. On the Properties of Neural Machine Translation: Encoder–Decoder Approaches. In *SSST-8 Workshop on Syntax, Semantics and Structure in Statistical Translation*. 103–111.

[3] 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 (2019).

[4] Shuyang Gao, Abhishek Sethi, Sanchit Agarwal, Tagyoung Chung, and Dilek Hakkani-Tur. 2019. Dialog State Tracking: A Neural Reading Comprehension Approach. In *Proceedings of the 20th annual meeting of the special interest group on discourse and dialogue*. 264–273.

[5] 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* (2019).

[6] Michael Heck, Carel van Niekerk, Nurul Lubis, Christian Geishauser, Hsien-Chin Lin, Marco Moresi, and Milica Gašić. 2020. TripPy: A Triple Copy Strategy for Value Independent Neural Dialog State Tracking. In *Proceedings of the 21th annual meeting of the special interest group on discourse and dialogue*. 35–44.

[7] Matthew Henderson, Blaise Thomson, and Jason D Williams. 2014. The second dialog state tracking challenge. In *Proceedings of the 15th annual meeting of the special interest group on discourse and dialogue*. 263–272.

[8] Ehsan Hosseini-Asl, Bryan McCann, Chien-Sheng Wu, Semih Yavuz, and Richard Socher. 2020. A simple language model for task-oriented dialogue. *arXiv preprint arXiv:2005.00796* (2020).

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

[10] Adarsh Kumar, Peter Ku, Anuj Kumar Goyal, Angeliki Metallinou, and Dilek Hakkani-Tur. 2020. MA-DST: Multi-Attention Based Scalable Dialog State Tracking. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*. 8107–8114.

[11] Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: Slot-Utterance Matching for Universal and Scalable Belief Tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 5478–5483.

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

[13] Lizi Liao, Long Le Hong, Yunshan Ma, Wenqiang Lei, and Tat-Seng Chua. 2020. Dialogue State Tracking with Incremental Reasoning. *Transactions of the Association for Computational Linguistics* (2020).

[14] Lizi Liao, Yunshan Ma, Xiangnan He, Richang Hong, and Tat-seng Chua. 2018. Knowledge-aware multimodal dialogue systems. In *Proceedings of the 26th ACM international conference on Multimedia*. 801–809.

[15] Lizi Liao, You Zhou, Yunshan Ma, Richang Hong, and Tat-seng Chua. 2018. Knowledge-aware Multimodal Fashion Chatbot. In *Proceedings of the 26th ACM international conference on Multimedia*. 1265–1266.

[16] Nikola Mrkšić, Diarmuid Ó Séaghdha, Tsung-Hsien Wen, Blaise Thomson, and Steve Young. 2017. Neural Belief Tracker: Data-Driven Dialogue State Tracking. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1777–1788.

[17] Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward Scalable Neural Dialogue State Tracking Model. *arXiv preprint arXiv:1812.00899* (2018).

[18] Osman Ramadan, Paweł Budzianowski, and Milica Gasic. 2018. Large-Scale Multi-Domain Belief Tracking with Knowledge Sharing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*. 432–437.

[19] Abhinav Rastogi, Dilek Hakkani-Tür, and Larry Heck. 2017. Scalable multi-domain dialogue state tracking. In *2017 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU)*. 561–568.

[20] Liliang Ren, Jianmo Ni, and Julian McAuley. 2019. Scalable and Accurate Dialogue State Tracking via Hierarchical Sequence Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing*. 1876–1885.

[21] Liliang Ren, Kaige Xie, Lu Chen, and Kai Yu. 2018. Towards Universal Dialogue State Tracking. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. 2780–2786.

[22] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108* (2019).

[23] Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get To The Point: Summarization with Pointer-Generator Networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 1073–1083.

[24] Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. 1715–1725.

[25] 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 *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. 3776–3783.

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

[27] 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 *Proceedings of the 14th annual meeting of the special interest group on discourse and dialogue*. 423–432.

[28] 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 *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics*. 438–449.

[29] Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. 2014. The dialog state tracking challenge series. *AI Magazine* 35, 4 (2014), 121–124.

[30] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. 2019. Transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771* (2019).

[31] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. 2019. Transferable Multi-Domain State Generator for Task-Oriented Dialogue Systems. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 808–819.

[32] Puyang Xu and Qi Hu. 2018. An End-to-end Approach for Handling Unknown Slot Values in Dialogue State Tracking. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*. 1448–1457.

[33] Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. 2019. Find or Classify? Dual Strategy for Slot-Value Predictions on Multi-Domain Dialog State Tracking. *arXiv* (2019).

[34] Zheng Zhang, Lizi Liao, Minlie Huang, Xiaoyan Zhu, and Tat-Seng Chua. 2019. Neural multimodal belief tracker with adaptive attention for dialogue systems. In *The World Wide Web Conference*. 2401–2412.

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

[36] Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *arXiv preprint arXiv:1911.06192* (2019).