Neural Dialogue State Tracking with Temporally Expressive Networks

Junfan Chen
BDBC and SKLSDE
Beihang University, China
chenjf@act.buaa.edu.cn

Richong Zhang*
BDBC and SKLSDE
Beihang University, China
zhangrc@act.buaa.edu.cn

Yongyi Mao
School of EECS
University of Ottawa, Canada
ymao@uottawa.ca

Jie Xu
School of Computing
University of Leeds, United Kingdom
j.xu@leeds.ac.uk

Abstract

Dialogue state tracking (DST) is an important part of a spoken dialogue system. Existing DST models either ignore temporal feature dependencies across dialogue turns or fail to explicitly model temporal state dependencies in a dialogue. In this work, we propose Temporally Expressive Networks (TEN) to jointly model the two types of temporal dependencies in DST. The TEN model utilizes the power of recurrent networks and probabilistic graphical models. Evaluating on standard datasets, TEN is demonstrated to improve the accuracy of turn-level-state prediction and the state aggregation.

1 Introduction

Spoken dialogue systems (SDS) connect users and computer applications through human-machine conversations. The users can achieve their goals, such as finding a restaurant, by interacting with a task-oriented SDS over multiple dialogue rounds or *turns*. Dialogue state tracking (DST) is an important task in SDS and the key function is to maintain the *state* of the system so as to track the progress of the dialogue. In the context of this work, a state (or aggregated state) is the user's intention or interest accumulated from the conversation history, and the user's intention or interest at each turn is referred to as turn-level state.

Many neural-network models have been successfully applied to DST. These models usually solve the DST problem by two approaches, the Implicit Tracking and the Explicit Tracking. As is shown in Figure 1 (a), the Implicit Tracking models (Henderson et al., 2014b,c; Mrksic et al., 2015; Ren et al., 2018; Ramadan et al., 2018; Lee et al., 2019) employs recurrent networks to accumulate features extracted from historical system action and user utterance pairs. A classifier is then built upon these accumulated features for state prediction. Although the Implicit Tracking captures temporal feature dependencies in recurrent-network cells, the state dependencies are not explicitly modeled. Only considering temporal feature dependencies is insufficient for accurate state prediction. This fact has been confirmed via an ablation study in our experiment.

Unlike the Implicit Tracking, the Explicit Tracking approaches, such as NBT (Mrksic et al., 2017) and GLAD (Zhong et al., 2018), model the state dependencies explicitly. From the model structure in Figure 1(b), the Explicit Tracking approaches first build a classifier to predict the turn-level state of each turn and then utilize a state aggregator for state aggregation.

Despite achieving remarkable improvements upon the previous models, current Explicit Tracking models can be further improved in two aspects. One is that the temporal feature dependencies should be considered in model design. The Explicit Tracking models only extract features from the current system action and user utterance pair. In practice, the slot-value pairs in different turns are highly dependent. For example, if a user specifies (`FOOD`, `italian`) at the current turn, he or she will probably not express it again in the future turns. For that reason, only extracting features from the current system action and user utterance pair is inadequate for turn-level state prediction.

The other is that the uncertainties in the state aggregation can be more expressively modeled. The state-aggregation approaches in current Explicit Tracking models are sub-optimal. The deterministic rule in GLAD will propagate errors to future turns and lead to incorrect state aggregation. The heuristic aggregation in NBT needs further estimate the best configuration of its coefficient . An

*Corresponding author

Findings of the Association for Computational Linguistics: EMNLP 2020, pages 1570–1579

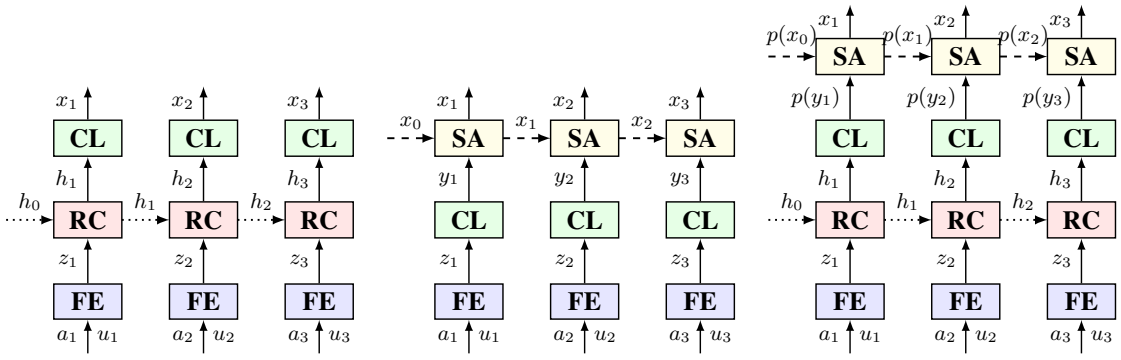

Figure 1: The model structures of Implicit Tracking, Explicit Tracking and Joint model. (a, u):the system action and user utterance. z: features extracted from the (a, u) pair. h:the hidden state of RNNs. y: the turn-level state. x: the aggregated state. **FE**:Feature Extractor, such as CNNs, RNNs. **RC**:Recurrent Cell, such as LSTM, GRU. **CL**:Classifier. **SA**:State Aggregator. The dotted arrowed lines emphasize modeling temporal feature dependencies. The dashed arrowed lines emphasize modeling temporal state dependencies.

approach that can both reduce the error propagation and require less parameter estimation is necessary for the state aggregation.

In this study, we propose a novel Temporally Expressive Networks (TEN) to jointly model the temporal feature dependencies and temporal state dependencies (Figure 1 (c)). Specifically, to improve the turn-level state prediction, we exploits hierarchical recurrent networks to capture temporal feature dependencies across dialogue turns. Furthermore, to reduce state aggregation errors, we introduce factor graphs to formulate the state dependencies, and employ belief propagation to handle the uncertainties in state aggregation. Evaluating on the DSTC2, WOZ and MultiWoZ datasets, TEN is shown to improve the accuracy of the turn-level state prediction and the state aggregation. The TEN model establishes itself as a new state-of-the-art model on the DSTC2 dataset and a state-of-the-art comparable model on the WOZ dataset.

2 Problem Statement

In a dialogue system, the state is represented as a set of *slot-value* pairs. Let $\mathcal{S}$ denote the predefined set of slots. For each slot $s \in \mathcal{S}$, let $\mathcal{V}(s)$ denote the set of all possible values associated with slot s. We also include an additional token, `unknown`, as a legal value for all slots to represent their value is not determined. And we define

$$\begin{aligned} \mathcal{V}^*(s) &:= \mathcal{V}(s) \cup \{\texttt{unknown}\} \\ \mathcal{V}^* &:= \bigcup_{s \in \mathcal{S}} \mathcal{V}^*(s) \end{aligned}$$

Let $\mathcal{X}$ denote the state space, and $x \in \mathcal{X}$ be a state configuration. Each state configuration x can be regarded as a function mapping $x(s)$ from $\mathcal{S}$ to $\mathcal{V}^*$. For example,

$$x(s) = \begin{cases} \texttt{italian}, & s = \texttt{FOOD} \\ \texttt{moderate}, & s = \texttt{PRICERANGE} \\ \texttt{unknown}, & s = \texttt{AREA} \end{cases} \quad (1)$$

Let x_t denotes the state configuration of the t^{th} dialogue turn, u_t denotes the user utterance of the t^{th} turn and a_t denotes the system action based on previous state x_{t-1}. Let $y_t \in \mathcal{X}$ be the turn-level state, which is meant to capture the user intention of the current utterance. The system computes the aggregated state x_t through a deterministic procedure, according to y_t and x_{t-1}. We next describe this procedure.

For any given s, we define an operator $\triangleleft$ on $\mathcal{V}^*(s)$ as follows. For any $v, v' \in \mathcal{V}^*(s)$,

$$v \triangleleft v' := \begin{cases} v, & \text{if } v' = \texttt{unknown} \\ v', & \text{otherwise} \end{cases} \quad (2)$$

We then extend the operator $\triangleleft$ to any two elements $x, y \in \mathcal{X}$, where $x \triangleleft y$ is also an element in $\mathcal{X}$

$$(x \triangleleft y)(s) := x(s) \triangleleft y(s). \quad (3)$$

Using this notation, the aggregation of states is precisely according to

$$x_t = x_{t-1} \triangleleft y_t. \quad (4)$$

For example, if x_{t-1} takes the configuration x in (1) and if y_t is

$$y_t(s) = \begin{cases} \texttt{chinese}, & s = \texttt{FOOD} \\ \texttt{unknown}, & s = \texttt{PRICERANGE} \\ \texttt{unknown}, & s = \texttt{AREA} \end{cases} \quad (5)$$

The aggregated state x_t is

$$x_t(s) = \begin{cases} \texttt{chinese}, & s = \texttt{FOOD} \\ \texttt{moderate}, & s = \texttt{PRICERANGE} \\ \texttt{unknown}, & s = \texttt{AREA} \end{cases} \quad (6)$$

The dialogue process can be characterized by a random process $\{(X_t, Y_t, A_t, U_t) : t = 1, 2, \ldots)\}$. In the DST problem, the probability measure $\mathbb{P}$ which defines the dialogue process is unknown. We are however given a set $\mathcal{R}$ of realizations drawn from $\mathbb{P}$, where each $r \in \mathcal{R}$ is a dialogue, given in the form of $\{(x_t^{(r)}, y_t^{(r)}, a_t^{(r)}, u_t^{(r)}) : t = 1, 2, \ldots)\}$. Let $x_{<t}$ denotes $(x_1, x_2, \ldots, x_t)$ and assume similar notations for $y_{<t}$, $a_{<t}$ etc. The learning problem for DST then becomes estimating $\mathbb{P}(x_t | a_{<t}, u_{<t})$ for every t.

3 Model

This section introduces the proposed TEN model, which consists of Action-Utterance Encoder, Hierarchical Encoder, Turn-level State Predictor and State Aggregator.

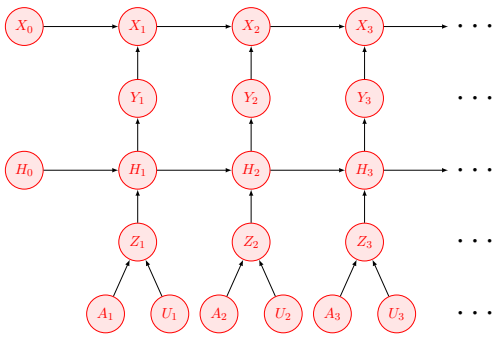

Figure 2: The probabilistic graphical model of TEN.

3.1 Model Structure

The overall model structure of TEN is shown in Figure 1 (c). we wish to express $\mathbb{P}(x_t | a_{<t}, u_{<t})$ using a probabilistic graphical model. For that purpose, we introduce two latent layers of random variables $\{H_t\}$ and $\{Z_t\}$, together with $\{Y_t\}$ and $\{X_t\}$, to form a Markov chain

$$\{(A_t, U_t)\} \rightarrow \{Z_t\} \rightarrow \{H_t\} \rightarrow \{Y_t\} \rightarrow \{X_t\}. \quad (7)$$

Then we can express the TEN model as a probabilistic graphical model shown in Figure 2. In the probabilistic graphical model, the variable Z_t is a matrix of size $K_{\mathrm{Z}} \times |\mathcal{S}|$, each column of $Z_t(s)$ corresponds to a slot $s \in \mathcal{S}$. Obtained from (A_t, U_t), Z_t is referred to as the "action-utterance encoding" at turn t which has a dimension of K_{Z}. The variable H_t is a matrix of size $K_{\mathrm{H}} \times |\mathcal{S}|$, with each column $H_t(s)$ also corresponding to the slot $s \in \mathcal{S}$. Here the recurrent $\{H_t\}$ layer is used to capture temporal feature dependencies, and H_t is referred to as the "hierarchical encoding", which has a dimension of K_{H}. In state aggregation, we introduce the factor graphs to model the state dependencies. The belief propagation is then employed to alleviate the error propagation. It allows the soft-label of Y_t and X_t keeping modeled. We next explain each module in detail.

3.1.1 Action-Utterance Encoder

This module's function is to summarize the input system action and user utterance to a unified representation. For later use, we first define a GRU-attention encoder or abbreviated as GAE. The GAE block first feeds an arbitrary-length sequence of word-embedding vectors $(\overline{w}_1, \overline{w}_2, ..., \overline{w}_n) := \overline{w}_{<n}$ to a GRU encoder and obtains a hidden state vector d_i at the i^{th} time step, then weighted-combine all the hidden-state vectors using attention mechanism to construct the output vector o. The computation process of the GAE block is

$$\begin{aligned} d_i &= \mathrm{GRU}\left(d_{i-1}, \overline{w}_i; \mathbf{W}\right) \\ o &= \sum_{i=1}^{n} \frac{\exp\left(d_i^T \cdot \theta\right)}{\sum_{j=1}^{n} \exp\left(d_j^T \cdot \theta\right)} d_i \end{aligned} \quad (8)$$

Here $\mathbf{W}$ is the parameter of the GRU networks and θ is the learnable parameter of attention mechanism. We simply introduce a notation $\mathrm{GAE}(\overline{w}_{<n}; \mathbf{W}, \theta)$ to indicate the above computation process (8) of the GAE block.

Utterance Encoder. Let $\overline{w}_{<n}^{\mathrm{u},t}$ denotes the word-embedding sequence of the t^{th} user utterance u_t. A GAE block is then used to obtain the utterance encoder with input $\overline{w}_{<n}^{\mathrm{u},t}$. For each slot $s \in \mathcal{S}$, an utterance encoding $\overline{u}_t(s)$ is computed by

$$\overline{u}_t(s) = \mathrm{GAE}(\overline{w}_{<n}^{\mathrm{u},t}; \mathbf{W}_{\mathrm{u}}, \theta_s) \quad (9)$$

Note that the GAEs for different slot s share the same parameter $\mathbf{W}_{\mathrm{u}}$, but they each have their own attention parameter θ_s.

Action Encoder. The system action at each turn may contain several phrases (Zhong et al., 2018). Suppose that action a_t contains m phrases. Each phrase $b_t^i \in a_t$ is then taken as a word sequence, and let its word-embedding sequence be denoted as $\overline{b}_t^i$. For each i and each slot s, $\overline{b}_t^i$ is passed to a GAE block and the action-phrase vector $c_t^i(s)$ is computed by

$$c_t^i(s) = \mathrm{GAE}(\overline{b}_t^i; \mathbf{W}_\mathrm{a}, \varphi_s) \tag{10}$$

Like utterance encoder, these $|\mathcal{S}|$ parallel GAE's share the same GRU parameter $\mathbf{W}_\mathrm{a}$ but each has its own attention parameters φ_s. Finally, we adopt the same approach proposed in (Zhong et al., 2018), which combines the action-phrase vectors to a single vector by attention mechanism. Specifically, the action encoding $\overline{a}_t(s)$ is obtained by interacting with utterance encoding $\overline{u}_t(s)$, calculated as

$$\overline{a}_t(s) = \sum_{i=1}^{m} \frac{\exp\left(\overline{u}_t(s)^T \cdot c_t^i(s)\right)}{\sum_{j=1}^{m} \exp\left(\overline{u}_t(s)^T \cdot c_t^j(s)\right)} c_t^i(s) \tag{11}$$

Action-utterance Encoding. The action-utterance encoding $z_t(s)$ is simply the concatenation of vectors $\overline{u}_t(s)$ and $\overline{a}_t(s)$.

3.1.2 Hierarchical Encoder

Instead of only utilizing the current action-utterance encoding for turn-level state prediction, in this module, we introduce the hierarchical recurrent networks to model the temporal feature dependencies across turns. Specifically, upon the GAE blocks, we use $|\mathcal{S}|$ parallel GRU networks to obtain the hierarchical encoding $\{h_t\}$ from all the historical action-utterance encoding vectors. The hierarchical encoding for each slot s is computed by

$$h_t(s) = \mathrm{GRU}\left(h_{t-1}(s), z_t(s); \mathbf{W}_\mathrm{h}\right) \tag{12}$$

where the parameter $\mathbf{W}_\mathrm{h}$ of these GRU networks, is shared across all slots.

3.1.3 Turn-level State Predictor

The Turn-level State Predictor is simply implemented by $|\mathcal{S}|$ softmax-classifiers, each for a slot s according to

$$\mathbb{P}\left(y_t(s)|a_{<t}(s), u_{<t}(s)\right) := \mathrm{smax}\left(\phi_s^T h_t(s)\right) \tag{13}$$

where smax denote the softmax function and ϕ_s with size $K_\mathrm{h} \times |\mathcal{V}^*(s)|$ serves as the weight matrix of the classifiers. We will denote this predictive distribution for turn-level state $y_t(s)$ computed by (13) as α_t^s.

3.1.4 State Aggregator

One of the insights in this work is that when a hard decision is made on the soft-label, the errors it creates may propagate to future turns, resulting in errors in future state aggregation. We insist that the soft-label of Y_t and X_t should be maintained so that the uncertainties in state aggregation can be kept in modeling. Thus we propose a state aggregator based on the factor graphs and handle these uncertainties using belief propagation.

Factor Graphs. For utilizing the factor graphs in state aggregation, we first introduce an indicator function, denoted by g, according to the deterministic aggregation rule $\triangleleft$. Specifically, for any $v, v', v'' \in \mathcal{V}^*(s)$,

$$g(v, v', v'') := \begin{cases} 1, & \text{if } v \triangleleft v' = v'' \\ 0, & \text{otherwise} \end{cases} \tag{14}$$

According to the probabilistic graphical model expressed in Figure 2, it can be derived that

$$\begin{aligned} &\mathbb{P}\quad (x_t|a_{<t}, u_{<t}) \\ &= \sum_{x_{<t-1}} \sum_{y_{<t}} \prod_{s\in\mathcal{S}} \alpha_t^s(y_t(s)) \prod_{\tau=1}^{t} g(x_{\tau-1}(s), y_\tau(s), x_\tau(s)) \\ &= \prod_{s\in\mathcal{S}} \sum_{x_{<t-1}(s)} \underbrace{\sum_{y_{<t}(s)} \underbrace{\alpha_t^s(y_t(s)) \prod_{\tau=1}^{t} g(x_{\tau-1}(s), y_\tau(s), x_\tau(s))}_{G(x_{<t}(s), y_{<t}(s))}}_{Q_t^s(x_t(s))} \end{aligned}$$

where the term $Q_t^s(x_t(s))$ above is precisely $\mathbb{P}(x_t(s)|a_{<t}, u_{<t})$, a distribution on $\mathcal{V}^*(s)$. It turns out that the term $G(x_{<t}(s), y_{<t}(s))$ in the double summation of $Q_t^s(x_t(s))$, despite its complexity, can be expressed elegantly using a factor graph in Figure 3.

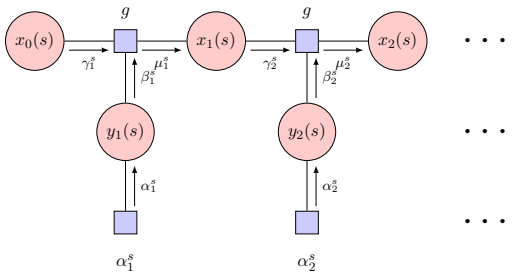

Figure 3: The factor graph for $G(x_{<t}(s), y_{<t}(s))$.

Belief Propagation. Factor graphs are powered by a highly efficient algorithm, called the belief

propagation or the sum-product algorithm, for computing the marginal distribution. In particular, the algorithm executes by passing "messages" along the edges of the factor graph and the sent message is computed from all incoming messages on its "upstream". For a detailed description of message computation rules in belief propagation, the reader is referred to (Kschischang et al., 2001).

Applying the principle of belief propagation, one can also efficiently express Q_t^s at each turn t for each slot s in terms of message passing. We now describe this precisely.

Let T denote the total number of turns of the dialogue. For each slot s, a factor graph representation $G(x_{<T}(s), y_{<T}(s))$ can be constructed. For each $t = 1, \ldots, T$, let messages β_t^s, γ_t^s and μ_t^s be introduced on the edges of the factor graph as shown in Figure 3 and the computation of these messages are given below.

$$\begin{cases} \beta_t^s & := & \alpha_t^s \\ \gamma_t^s & := & \mu_{t-1}^s \\ \mu_t^s(v) & := & \sum\limits_{(v',v'')\in\mathcal{V}^*(s)\times\mathcal{V}^*(s)} g(v',v'',v)\gamma_t^s(v')\beta_t^s(v'') \end{cases} \tag{15}$$

where μ_0^s is defined by

$$\mu_0^s(v) = \begin{cases} 1, & \text{if } v = \texttt{unknown} \\ 0, & \text{otherwise.} \end{cases}$$

According to message computation rule given in (15), for each $t \leq T$ and each slot $s \in \mathcal{S}$, $\mu_t^s = Q_t^s$. Recalling that Q_t^s is the predictive distribution for state $x_t(s)$ and α_t^s is the predictive distribution for turn-level state $y_t(s)$, we have completed specifying how the factor graphs and the belief propagation are utilized for state aggregation.

3.2 Loss Function and Training

Under the TEN model, the cross-entropy loss on the training set $\mathcal{R}$ follows the standard definition as below

$$\mathcal{L}_{\text{TEN}} := \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{S}}\sum_{t=1}^{T^{(r)}} -\log Q_t^s(x_t^{(r)}(s)) \tag{16}$$

where the superscript "(r)" indexes a training dialogue in $\mathcal{R}$. It is worth noting that this loss function, involving the message computation rules, can be directly optimized by the stochastic gradient descent (SGD) method.

For ablation studies, we next present three ablated versions of the TEN model.

TEN–Y Model In this model, we discard the $\{Y_t\}$ layer of TEN (hence the name TEN–Y) and conduct state aggregation using RNNs. The model then turns to be an Implicit Tracking model. The state distribution $\mathbb{P}(x_t(s)|a_{<t}, u_{<t})$ is computed directly by the softmax-classifiers in (13). We will denote the state distribution computed this way by $\widetilde{Q}_t^s$. The cross-entropy loss is then defined as

$$\mathcal{L}_{\text{TEN-Y}} := \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{S}}\sum_{t=1}^{T^{(r)}} -\log \widetilde{Q}_t^s(x_t^{(r)}(s)) \tag{17}$$

TEN–X Model In this model, instead of training against the state sequence $\{x_t\}$, the training target is taken as the corresponding turn-level state sequence $\{y_t\}$. The computation of $\{x_t\}$ can be done through the operator $\triangleleft$: $x_t = x_{t-1} \triangleleft y_t$. When using the turn-level state as training target, one discards the $\{X_t\}$ layer of TEN (hence the name TEN–X). The difference between TEN–X and TEN is that TEN–X aggregate states using the deterministic rule $\triangleleft$ while TEN using the factor graphs. The cross-entropy loss for TEN–X is naturally defined as

$$\mathcal{L}_{\text{TEN-X}} := \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{S}}\sum_{t=1}^{T^{(r)}} -\log \alpha_t^s(y_t^{(r)}(s)) \tag{18}$$

TEN–XH Model In this model, the Hierarchical Encoder layer $\{H_t\}$ is removed from TEN–X, and the model is reduced to an Explicit Tracking mode. In this case, the computation of α_t^s (or $\mathbb{P}(y_t(s)|a_{<t}, \widetilde{u}_{<t})$) in (13) is done by replacing the input $h_t(s)$ with the action-utterance encoding $z_t(s)$. We will denote the α_t^s computed this way by $\widetilde{\alpha}_t^s$. The TEN–XH and TEN–X models are different in whether the temporal feature dependencies are considered or not. The cross-entropy loss for TEN–XH is

$$\mathcal{L}_{\text{TEN-XH}} := \sum_{r\in\mathcal{R}}\sum_{s\in\mathcal{S}}\sum_{t=1}^{T^{(r)}} -\log \widetilde{\alpha}_t^s(y_t^{(r)}(s)) \tag{19}$$

4 Experiment

4.1 Datasets

The second Dialogue State Tracking Challenge dataset (DSTC2) (Henderson et al., 2014a), the second version of the Wizard-of-Oz dataset (WOZ) (Rojas-Barahona et al., 2017) and

MultiDomain Wizard-of-Oz dataset (MultiWOZ) (Budzianowski et al., 2018) are used to evaluate the models. Both the DSTC2 and WOZ datasets contain conversations between users and task-oriented dialogue systems about finding suitable restaurants around Cambridge. The DSTC2 and WOZ datasets share the same ontology, which contain three informable slots: `FOOD`, `AREA`, `PRICERANGE`. The official DSTC2 dataset contains some spelling errors in the user utterances, as is pointed out in (Mrksic et al., 2017). Thus we use the manually corrected version provided by (Mrksic et al., 2017). This dataset consists of $3,235$ dialogues with $25,501$ turns. There are $1,612$ dialogues for training, 506 dialogues for validation and $1,117$ dialogues for testing. The average turns per dialogue is 14.49. In the WOZ dataset, there are $1,200$ dialogues with $5,012$ turns. The number of dialogues used for training, validation and testing are 600, 200 and 400 respectively. The average turns per dialogue is 4. The MultiWOZ dataset is a large multi-domain dialogue state tracking dataset with 30 slots, collected from human-human conversations. The training set contains $8,438$ dialogues with $115,424$ turns. There are respectively $1,000$ dialogues in validation and test set. The average turns per dialogue is 13.68.

4.2 Evaluation Metrics and Compared Models

In this work, we focus on the standard evaluation metrics, *joint goal accuracy*, which is described in (Henderson et al., 2014a). The *joint goal accuracy* is the proportion of dialogue turns whose states are correctly predicted. In addition, we also report the *turn-level state accuracy* of TEN–XH and TEN–X model for ablation studies.

The models used for comparison include NBT-DNN (Mrksic et al., 2017), NBT-CNN (Mrksic et al., 2017), Scalable (Rastogi et al., 2017), MemN2N (Liu and Perez, 2017), PtrNet (Xu and Hu, 2018), LargeScale (Ramadan et al., 2018), GLAD (Ramadan et al., 2018), GCE (Nouri and Hosseini-Asl, 2018), StateNetPSI (Ren et al., 2018), SUMBT (Lee et al., 2019), HyST (Goel et al., 2019), DSTRead+JST (Gao et al., 2019), TRADE (Wu et al., 2019), COMER (Ren et al., 2019), DSTQA (Zhou and Small, 2019), MERET (Huang et al., 2020) and SST (Chen et al., 2020).

Table 1: Joint goal accuracy on the DSTC2, WOZ and MultiWOZ dataset.

Model	DSTC2	WOZ	MultiWOZ
NBT-DNN	72.6	84.4	-
NBT-CNN	73.4	84.2	-
Scalable	70.3	-	-
MemN2N	74.0	-	-
PtrNet	72.1	-	-
LargeScale	-	85.5	25.8
GLAD	74.5	88.1	35.6
GCE	-	88.5	35.6
StateNetPSI	75.5	88.9	-
SUMBT	-	**91.0**	42.4
HyST	-	-	44.2
DSTRead+JST	-	-	47.3
TRADE	-	-	48.6
COMER	-	88.6	45.7
DSTQA	-	-	**51.4**
MERET	-	-	50.9
SST	-	-	51.2
TEN–XH	73.5	88.8	42.0
TEN–Y	74.7	89.6	45.9
TEN–X	76.2	89.3	46.3
TEN	**77.3**	90.8	46.6

4.3 Implementation

The proposed models are implemented using the Pytorch framework. The code and data are released on the Github page[1]. The word embedding is the concatenation of the pre-trained GloVe embeddings (Pennington et al., 2014) and the character n-gram embeddings (Hashimoto et al., 2017). We tune the hyper-parameters by grid search on the validation set. The GAE block is implemented with bi-directional GRUs, and the hidden state dimension of the GAE is 50. The hidden state dimension of the GRU used in the Hierarchical Encoder module is 50. The fixed learning rate is 0.001. The Adam optimizer (Kingma and Ba, 2015) with the default setting is used to optimize the models. It is worth mentioning that the TEN model can be difficult to train with SGD from a cold start. This is arguably due to the "hard" g function. That is, the $\{0, 1\}$-valued nature of g is expected to result in sharp barriers in the loss landscape, preventing gradient-based optimization to cross. Thus when training TEN, we start with the parameters obtained from a pre-trained TEN–X model.

[1]https://github.com/BDBC-KG-NLP/TEN_EMNLP2020

4.4 Evaluation Results

The joint goal accuracy results on the DSTC2,WOZ and MultiWOZ datasets are shown in Table 1. From the table, we observe that the proposed TEN model outperforms previous models on both DSTC2 and WOZ datasets, except SUMBT, a model boosted with pre-trained BERT (Devlin et al., 2019) model. It is worth noting that TEN, built upon attention-based GRU encoders, achieves comparable performance with SUMBT, without incorporating pre-trained language models. This fact demonstrates that TEN is a strong model for DST. Comparing to TEN–XH, the TEN–X model obtains impressive 2.7%, 0.5% and 4.3% performance gains on the DSTC2, WOZ and MultiWOZ dataset respectively. These performance gains demonstrate that the state estimation benefits from more accurate turn-level state prediction. The TEN model further improves upon the TEN–X model by 1.1% on the DSTC2 dataset, 1.5% on the WOZ dataset and 0.3% on the MultiWOZ dataset. The TEN model achieves these improvements by modeling uncertainties with the belief propagation in the state aggregation. Although both TEN–Y and TEN have modeled the temporal feature dependencies, TEN–Y performs much worse than TEN. This fact indicates that only considering temporal feature dependencies is inadequate for DST. Models relying on pre-defined ontologies (including GLAD,GCE,SUMBT and TEN) suffer from computational complexity when applying to multi-domain DST datasets with a large set of slots (Ren et al., 2019), which leads to worse performance than recent generation-based models (DSTRead+JST,TRADE,DSTQA,MERET and SST, specially designed for multi-domain DST) on the MultiWOZ dataset.

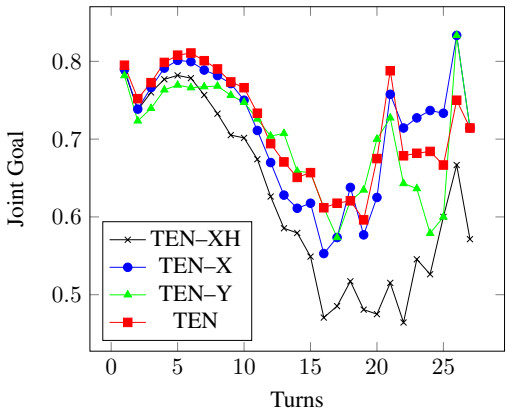

Figure 4: Temporal analysis on the DSTC2 dataset.

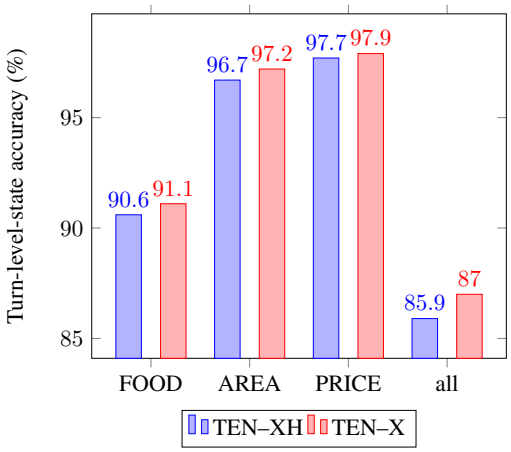

Figure 5: turn-level state accuracy for TEN–XH and TEN–X on the DSTC2 dataset. The *PRICE* indicates the PRICERANGE slot. The *all* denotes the proportion of dialogue turns that the turn-level states for all slots are correctly predicted.

4.5 Temporal Analysis

To analyze how the temporal dependencies influence the state tracking performance, we report the joint goal accuracy at each dialogue turn on the DSTC2 dataset. As shown in Figure 4, the joint goal accuracy of proposed models generally decrease at earlier turns and increase at later turns, as the turns increase. This phenomenon can be explained by the fact: in the earlier stage of the dialogue, more slots are involved in the conversation as the dialogue progress; thus more slot-value pairs need to be predicted in state estimation, making the state harder to calculate correctly; in the later stage of dialogue, the state becomes fixed because the values for all slots are already determined, making the state easier to predict. Another observation is that the gaps between TEN–XH and TEN generally increase as the turns increase, showing that modeling temporal dependencies reduces state estimation errors, especially when the dialogue is long. By modeling temporal feature dependencies and temporal state dependencies respectively, TEN–Y and TEN–X also perform better than TEN–XH as the turns increase.

4.6 Effectiveness of the Hierarchical Encoder

To prove the effectiveness of the Hierarchical Encoder module, we report the turn-level state accuracy for TEN–XH and TEN–X on the DSTC2 dataset. From the results in Figure 5, we observe that TEN–X, with the Hierarchical Encoder module, achieves higher turn-level state accuracy than

Table 2: An example of dialogue state tracking. We only report the results from turn 1 to turn 4 on slot s = FOOD and focus on dontcare(dcr) and unknown(unk) value due to space limitation. **S** and **U** represent the system utterance and the user utterance, respectively. The boldface emphasizes the highest-probability value.

t	(a_t, u_t)	α_t^s	$y_t(s)$	Q_t^s	x_t^s	TEN–X	TEN
1	**S**:welcome to cambridge restaurant system. **U**:im looking for a moderately priced	(dcr, 0.00) (**unk**, **0.99**)	unk	(dcr, 0.00) (**unk**, **0.99**)	unk	unk	unk
2	**S**:moderate price range. what type of food do you want? **U**:restaurant and it should be	(dcr, 0.00) (**unk**, **0.48**)	unk	(dcr, 0.00) (**unk**, **0.48**)	unk	unk	unk
3	**S**:you want a restaurant serving any type of food right? **U**:yea	(dcr, 0.45) (**unk**, **0.54**)	dcr	(**dcr**, **0.45**) (unk, 0.26)	dcr	unk	dcr
4	**S**:what part of town do you have in mind? **U**:north	(dcr, 0.00) (**unk**, **0.99**)	unk	(**dcr**, **0.45**) (unk, 0.26)	dcr	unk	dcr

TEN–XH for all slots.

Recall that TEN–X achieves higher joint goal accuracy than TEN–XH, we could think that the performance gain for TEN–X is due to its improvement in turn-level state prediction. This fact demonstrates the significance of considering temporal feature dependencies in turn-level state prediction and illustrates the effectiveness of the Hierarchical Encoder module in TEN–X.

4.7 Effectiveness of the Belief Propagation

Table 2 is an example of dialogue state tracking selected from the test set of the DSTC2 dataset. As we observe from the table, at turn 1 and turn 2, the user does not specify any food type; both TEN–X and TEN correctly predict the true value unknown. At turn 3, the user expresses that he or she does not care about the food type. This time the turn-level state predictor gets an incorrect turn-level state value unknown, instead of the correct one dontcare. Thus TEN–X gets a wrongly aggregated state value unknown with aggregating rule $\triangleleft$. On the contrary, TEN can still correctly obtain the correct state with the belief propagation, in spite of the wrong turn-level state. At turn 4, the turn-level state predictor easily predicts the correct value unknown and TEN keeps the state correct. But TEN–X fails to obtain the correct state again because of the wrong decision made at the last turn. This example shows the effectiveness and robustness of the state aggregation approach equipped with the belief propagation.

5 Related Works

Traditional works deal with the DST task using Spoken Language Understanding (SLU), including (Thomson and Young, 2010; Wang and Lemon, 2013; Lee and Kim, 2016; Liu and Perez, 2017; Jang et al., 2016; Shi et al., 2016; Vodolán et al., 2017). Joint modeling of SLU and DST (Henderson et al., 2014c; Zilka and Jurcícek, 2015; Mrksic et al., 2015) has also been presented and shown to outperform the separate SLU models. Models like (Sun et al., 2014; Yu et al., 2015) incorporate statistical semantic parser for modeling the dialogue context. These models rely on hand-crafted features or delexicalisation strategies and are difficult to scale to realistic applications.

Recently, neural network models have been applied in the DST task, and there are mainly two model design approaches. One approach aggregates the features extracted from previous turns of the dialogue using recurrent neural networks, including StateNet (Ren et al., 2018), LargeScale(Ramadan et al., 2018) and SUMBT (Lee et al., 2019). The other approach, like NBT (Mrksic et al., 2017) and GLAD (Zhong et al., 2018), build a model for predicting turn-level state, and estimate the state by accumulating all previous turn-level states. The design of TEN integrates the advantages of both approaches.

Another topic related to our work is the Markov decision process (MDP) and the factor graphs. Several works define a dialogue system as a partially observable Markov decision process (POMDP), including (Williams and Young, 2007; Thomson and Young, 2010; Gasic and Young, 2011; Yu et al., 2015). In this paper, the definition of the dialogue process is related to the Markov decision process. The factor graphs have been applied in many applications, such as social influence analysis (Tang et al., 2009), knowledge base alignment (Wang et al., 2012), entity linking (Ran et al., 2018) and visual dialog generation (Schwartz et al., 2019). The factor graphs in these applications are used to integrate different sources of features or repre-

sentations into a unified probabilistic model. In this paper, the factor graphs are naturally adopted to tackle the error propagation problem in state aggregation.

6 Concluding Remarks

Our inspiration for TEN comes from a careful study of the dialogue process. This allows us to lay out the dependency structure of the network as in Figure 1 (c), where the temporal feature dependencies and the temporal state dependencies are jointly modelled. The application of the belief propagation in this model allows an elegant combination of graphical models with deep neural networks. The proposed model may generalize to other sequence prediction tasks.

Acknowledgment

This work is supported partly by the National Natural Science Foundation of China (No. 61772059, 61421003), by the Beijing Advanced Innovation Center for Big Data and Brain Computing (BDBC), by the Fundamental Research Funds for the Central Universities, by the Beijing S&T Committee (No. Z191100008619007) and by the State Key Laboratory of Software Development Environment (No. SKLSDE-2020ZX-14).

References

Pawel Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. 2018. Multiwoz - A large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP 2018*, pages 5016–5026.

Lu Chen, Boer Lv, Chi Wang, Su Zhu, Bowen Tan, and Kai Yu. 2020. Schema-guided multi-domain dialogue state tracking with graph attention neural networks. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 7521–7528. AAAI Press.

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

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

Milica Gasic and Steve J. Young. 2011. Effective handling of dialogue state in the hidden information state pomdp-based dialogue manager. *ACM Trans. Speech Lang. Process.*, 7(3):4:1–4:28.

Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. 2019. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. In *Interspeech 2019*, pages 1458–1462.

Kazuma Hashimoto, Caiming Xiong, Yoshimasa Tsuruoka, and Richard Socher. 2017. A joint many-task model: Growing a neural network for multiple NLP tasks. In *EMNLP 2017*, pages 1923–1933.

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

Matthew Henderson, Blaise Thomson, and Steve J. Young. 2014b. Robust dialog state tracking using delexicalised recurrent neural networks and unsupervised adaptation. In *SLT 2014*, pages 360–365.

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

Yi Huang, Junlan Feng, Min Hu, Xiaoting Wu, Xiaoyu Du, and Shuo Ma. 2020. Meta-reinforced multi-domain state generator for dialogue systems. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 7109–7118. Association for Computational Linguistics.

Youngsoo Jang, Jiyeon Ham, Byung-Jun Lee, Youngjae Chang, and Kee-Eung Kim. 2016. Neural dialog state tracker for large ontologies by attention mechanism. In *SLT 2016*, pages 531–537.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR 2015*.

Frank R. Kschischang, Brendan J. Frey, and Hans-Andrea Loeliger. 2001. Factor graphs and the sum-product algorithm. *IEEE Trans. Information Theory*, 47(2):498–519.

Byung-Jun Lee and Kee-Eung Kim. 2016. Dialog history construction with long-short term memory for robust generative dialog state tracking. *D&D*, 7(3):47–64.

Hwaran Lee, Jinsik Lee, and Tae-Yoon Kim. 2019. SUMBT: slot-utterance matching for universal and scalable belief tracking. In *ACL 2019*, pages 5478–5483.

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

Nikola Mrksic, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-hao Su, David Vandyke, Tsung-Hsien Wen, and Steve J. Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. In *ACL 2015*, pages 794–799.

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

Elnaz Nouri and Ehsan Hosseini-Asl. 2018. Toward scalable neural dialogue state tracking model. *CoRR*, abs/1812.00899.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *EMNLP 2014*, pages 1532–1543.

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

Chenwei Ran, Wei Shen, and Jianyong Wang. 2018. An attention factor graph model for tweet entity linking. In *WWW 2018*, pages 1135–1144.

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

Liliang Ren, Jianmo Ni, and Julian J. McAuley. 2019. Scalable and accurate dialogue state tracking via hierarchical sequence generation. In *EMNLP 2019*, pages 1876–1885.

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

Lina Maria Rojas-Barahona, Milica Gasic, Nikola Mrksic, Pei-Hao Su, Stefan Ultes, Tsung-Hsien Wen, Steve J. Young, and David Vandyke. 2017. A network-based end-to-end trainable task-oriented dialogue system. In *EACL 2017*, pages 438–449.

Idan Schwartz, Seunghak Yu, Tamir Hazan, and Alexander G. Schwing. 2019. Factor graph attention. In *CVPR 2019*, pages 2039–2048.

Hongjie Shi, Takashi Ushio, Mitsuru Endo, Katsuyoshi Yamagami, and Noriaki Horii. 2016. Convolutional neural networks for multi-topic dialog state tracking. In *IWSDS 2016*, pages 451–463.

Kai Sun, Lu Chen, Su Zhu, and Kai Yu. 2014. The SJTU system for dialog state tracking challenge 2. In *SIGDIAL Conference*, pages 318–326. The Association for Computer Linguistics.

Jie Tang, Jimeng Sun, Chi Wang, and Zi Yang. 2009. Social influence analysis in large-scale networks. In *SIGKDD 2009*, pages 807–816.

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

Miroslav Vodolán, Rudolf Kadlec, and Jan Kleindienst. 2017. Hybrid dialog state tracker with ASR features. In *EACL 2017*, pages 205–210.

Zhichun Wang, Juanzi Li, Zhigang Wang, and Jie Tang. 2012. Cross-lingual knowledge linking across wiki knowledge bases. In *WWW 2012*, pages 459–468.

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 2013*, pages 423–432.

Jason D. Williams and Steve J. Young. 2007. Partially observable markov decision processes for spoken dialog systems. *Comput. Speech Lang.*, 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 2019*, pages 808–819.

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

Kai Yu, Kai Sun, Lu Chen, and Su Zhu. 2015. Constrained markov bayesian polynomial for efficient dialogue state tracking. *IEEE/ACM Trans. Audio, Speech & Language Processing*, 23(12):2177–2188.

Victor Zhong, Caiming Xiong, and Richard Socher. 2018. Global-locally self-attentive dialogue state tracker. In *ACL 2018*.

Li Zhou and Kevin Small. 2019. Multi-domain dialogue state tracking as dynamic knowledge graph enhanced question answering. *CoRR*, abs/1911.06192.

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