Unsupervised Dialog Structure Learning

Weiyan Shi
University of California, Davis
wyshi@ucdavis.edu

Tiancheng Zhao
Carnegie Mellon University
tianchez@cs.cmu.edu

Zhou Yu
University of California, Davis
joyu@ucdavis.edu

Abstract

Learning a shared dialog structure from a set of task-oriented dialogs is an important challenge in computational linguistics. The learned dialog structure can shed light on how to analyze human dialogs, and more importantly contribute to the design and evaluation of dialog systems. We propose to extract dialog structures using a modified VRNN model with discrete latent vectors. Different from existing HMM-based models, our model is based on variational-autoencoder (VAE). Such model is able to capture more dynamics in dialogs beyond the surface forms of the language. We find that qualitatively, our method extracts meaningful dialog structure, and quantitatively, outperforms previous models on the ability to predict unseen data. We further evaluate the model's effectiveness in a downstream task, the dialog system building task. Experiments show that, by integrating the learned dialog structure into the reward function design, the model converges faster and to a better outcome in a reinforcement learning setting.

1 Introduction

Human dialogs are like well-structured buildings, with words as the bricks, sentences as the floors, and topic transitions as the stairs connecting the whole building. Therefore, discovering dialog structure is crucial for various areas in computational linguistics, such as dialog system building (Young, 2006), discourse analysis (Grosz and Sidner, 1986), and dialog summarization (Murray et al., 2005; Liu et al., 2010). In domain specific tasks such as restaurant booking, it's common for people to follow a typical conversation flow. Current dialog systems require human experts to design the dialog structure, which is time consuming and sometimes insufficient to satisfy various customer needs. Therefore, it's of great importance to automatically discover dialog structure from existing human-human conversations and incorporate it into the dialog system design.

However, modeling human conversation is not easy for machines. Some previous work rely on human annotations to learn dialog structures in supervised learning settings (Jurafsky, 1997). But since human labeling is expensive and hard to obtain, such method is constrained by the small size of training examples, and by the limited number of application domains (Zhai and Williams, 2014). Moreover, structure annotations on human conversation can be subjective, which makes it hard to reach inter-rater agreements. Therefore, we propose an unsupervised method to infer the latent dialog structure since unsupervised methods do not require annotated dialog corpus.

Limited previous work has studied unsupervised methods to model the latent dialog structure. Most of the previous methods use the *hidden Markov model* to capture the temporal dependency within human dialogs (Chotimongkol, 2008; Ritter et al., 2010; Zhai and Williams, 2014). We propose to adopt a new type of models, the *variational recurrent neural network* (VRNN, a recurrent version of the VAE) (Chung et al., 2015), and infer the latent dialog structure with variational inference. VRNN is suitable for modeling sequential information. Compared to the simpler HMM models, VRNN also has the flexibility to model highly non-linear dynamics (Chung et al., 2015) in human dialogs.

Our basic approach assumes that the dialog structure is composed of a sequence of latent states. Each conversational exchange (a pair of user and system utterances at time t) belongs to a *latent state*, which has causal effect on the future latent states and the words the conversants produce. Because discrete latent states are more interpretable than continuous ones, we combine

VRNN with Gumbel-Softmax (Jang et al., 2016) to obtain discrete latent vectors to represent the latent states. A common way to represent the dialog structure both visually and numerically is to construct a transition probability table among latent states. The idea of transition table inspires us to develop two model variants to model the dependency between states indirectly and directly.

Once we obtain such a human-readable dialog structure, we can use it to facilitate many downstream tasks, such as dialog system training. The motivation is that the dialog structure contains important information on the flow of the conversation; if the automatic dialog system can mimic the behaviour in human-human dialogs, it can interact with users in a more natural and user-friendly way. Therefore, we propose to integrate the dialog structure information into the reward design of the reinforcement learning (RL). Experiments show that the model with the proposed reward functions converges faster to a better success rate.

2 Related Work

Variational Autoencoders (VAEs) (Kingma and Welling, 2013; Doersch, 2016; Kingma et al., 2014) have gained popularity in many computational linguistics tasks due to its interpretable generative model structure (Miao and Blunsom, 2016; Miao et al., 2017). Zhao et al. (2018) applied VAE to learn discrete sentence representations and achieved good results. This is similar to our work, but we focus more on modeling the dialog structure and using the learned structure to improve the dialog system training. Serban et al. (2017) presented a VHRED model which combines the VRNN and encoder-decoder structure for direct dialog response generation. While also similar, our model uses discrete latent vectors instead of continuous ones, and re-constructs the utterances to recover the latent dialog structure, instead of modeling the responses directly.

There are some previous studies on discovering latent structure of conversations (Chotimongkol, 2008; Ritter et al., 2010; Zhai and Williams, 2014). But they are all based on Hidden Markov Model (HMM). Ritter et al. (2010) extended the HMM-based method in Chotimongkol (2008) by adding additional word sources to social interaction data on Twitter. Zhai and Williams (2014) decoupled the number of topics and the number of states to allow an additional layer of information in task-oriented dialogs. Our work also focuses on task-oriented dialogs but adopts the VRNN to perform variational inference in the model. According to Chung et al. (2015), the VRNN retains the flexibility to model highly non-linear dynamics, compared to simpler Dynamic Bayesian Network models such as HMM.

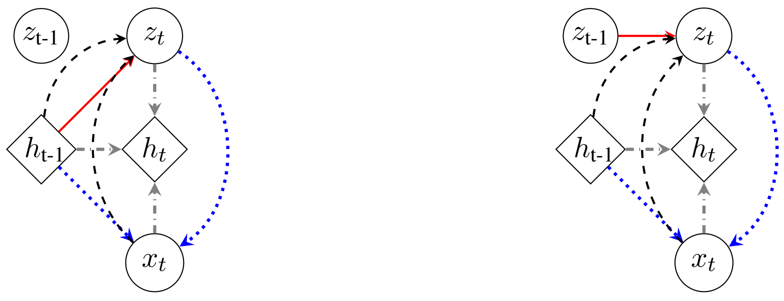
Gunasekara et al. (2017) described a Quantized-Dialog Language Model (QDLM) for task-oriented dialog systems, which performs clustering on utterances and models the dialog as a sequence of clusters to predict future responses. The idea of dialog discretization is similar to our method, but we choose VAE over simple clustering to allow more context-sensitivity and to capture more dynamics in the dialog beyond surface forms of the conversation. Additionally, we propose to utilize the dialog structure information to improve the dialog system training.

Traditional reward functions in RL dialog training use delayed reward to provide feedback to the model. However, delayed reward suffers from potential slow convergence rate problem, so some studies integrated estimated per-turn immediate reward. For example, Ferreira and Lefèvre (2013) studied expert-based reward shaping in dialog management. We use the KL-divergence between the transition probabilities and the predicted probabilities as the immediate per-turn reward. Different from the expert-based reward shaping, such reward does not require any manual labels and is generalizable to different tasks.

3 Models

Fig. 1 gives an overview of the Discrete-VRNN (D-VRNN) model and the Direct-Discrete-VRNN (DD-VRNN) model. In principal, the VRNN contains a VAE at every timestep, and these VAEs are connected by a state-level RNN. The hidden state variable $\mathbf{h}_{t\text{-}1}$ in this RNN encodes the dialog context up to time t. This connection helps the VRNN to model the temporal structure of the dialog (Chung et al., 2015). The observed inputs $\mathbf{x}_t$ to the model is the constructed utterance embeddings. $\mathbf{z}_t$ is the latent vector in the VRNN at time t. Different from Chung et al. (2015), $\mathbf{z}_t$ in our model is a discrete one-hot vector of dimension N, where N is the total number of latent states.

The major difference between D-VRNN and DD-VRNN lies in the priors of $\mathbf{z}_t$. In D-VRNN, we assume that $\mathbf{z}_t$ depends on the entire dialog



(a) Discrete-VRNN (b) Direct-Discrete-VRNN

Figure 1: Discrete-VRNN (D-VRNN) and Direct-Discrete-VRNN (DD-VRNN) overview. D-VRNN and DD-VRNN use different priors to model the transition between $\mathbf{z}_t$, shown in red solid lines. The regeneration of $\mathbf{x}_t$ is in blue dotted lines. The recurrence of the state-level RNN is in gray dash-dotted lines. The inference of $\mathbf{z}_t$ is in black dashed lines.

context $\mathbf{h}_{t\text{-}1}$, shown in red in Fig. 1(a), which is the same as in Chung et al. (2015); while in DD-VRNN we assume that in the prior, $\mathbf{z}_t$ directly depends on $\mathbf{z}_{t\text{-}1}$ in order to model the direct transition between different latent states, shown in red in Fig. 1(b). We use $\mathbf{z}_t$ and $\mathbf{h}_{t\text{-}1}$ to regenerate the current utterances $\mathbf{x}_t$ instead of generating the next utterances $\mathbf{x}_{t+1}$, shown in blue dotted lines in Fig. 1. The idea of regeneration helps recover the dialog structure. Next, the recurrence in the RNN takes $\mathbf{h}_{t\text{-}1}$, $\mathbf{x}_t$ and $\mathbf{z}_t$ to update itself, and allows the context to be passed down as the dialog proceeds, shown in gray dash-dotted lines. Finally in the inference, we construct the posterior of $\mathbf{z}_t$ with the context $\mathbf{h}_{t-1}$ and $\mathbf{x}_t$, and infer $\mathbf{z}_t$ by sampling from the posterior, shown in black dashed lines in Fig. 1. The mathematical details of each operation are described below. $\varphi_\tau^{(\cdot)}$ are highly flexible feature extraction functions such as neural networks. $\varphi_\tau^{\mathbf{x}}$, $\varphi_\tau^{\mathbf{z}}$, $\varphi_\tau^{\text{prior}}$, $\varphi_\tau^{\text{enc}}$, $\varphi_\tau^{\text{dec}}$ are feature extraction networks for the input $\mathbf{x}$, the latent vector $\mathbf{z}$, the prior, the encoder and the decoder.

Sentence Embedding. $u_t = [\mathrm{w}_{1,t}, \mathrm{w}_{2,t}, ...\mathrm{w}_{\mathrm{n_w},t}]$ and $s_t = [\mathrm{v}_{1,t}, \mathrm{v}_{2,t}, ...\mathrm{v}_{\mathrm{n_v},t}]$ are the user utterance and the system utterance at time t, where $\mathrm{w}_{\mathrm{i,j}}$ and $\mathrm{v}_{\mathrm{i,j}}$ are individual words. The concatenation of utterances from both parties, $x_t = [u_t, s_t]$, is the observed variable in the VAE. We use Mikolov et al. (2013) to perform word embedding and the average of the word embedding vectors of u_t and s_t are $\overline{u_t}$ and $\overline{s_t}$. The concatenation of $\overline{u_t}$ and $\overline{s_t}$ is used as the feature extraction of $\mathbf{x}_t$, namely $\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t) = [\overline{u_t}, \overline{s_t}]$. $\varphi_\tau^{\mathbf{x}}(\mathbf{x}_t)$ is the model inputs.

Prior in D-VRNN. The prior quantifies our assumption on $\mathbf{z}_t$ before we observe the data. In the D-VRNN, it's reasonable to assume that the prior of $\mathbf{z}_t$ depends on the context $\mathbf{h}_{t\text{-}1}$ and follows the distribution shown in Eq. (1), because conversation context is a critical factor that influences dialog transitions. Since $\mathbf{z}_t$ is discrete, we use softmax to obtain the distribution.

$$\mathbf{z}_t \sim \text{softmax}(\varphi_\tau^{\text{prior}}(\mathbf{h}_{t-1})) \tag{1}$$

Prior in DD-VRNN. The dependency of $\mathbf{z}_t$ on the entire context $\mathbf{h}_{t\text{-}1}$ in Eq. (1) makes it difficult to disentangle the relation between $\mathbf{z}_{t\text{-}1}$ and $\mathbf{z}_t$. But this relation is crucial in decoding how conversations flow from one conversational exchange to the next one. So in DD-VRNN, we directly model the influence of $\mathbf{z}_{t\text{-}1}$ on $\mathbf{z}_t$ in the prior, shown in Eq. (2) and Fig. 1(b). To fit this prior distribution into the variational inference framework, we approximate $p(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t})$ with $p(\mathbf{z}_t|\mathbf{z}_{t\text{-}1})$ in Eq. (3). Later, we show that the designed new prior has benefits under certain scenarios.

$$\mathbf{z}_t \sim \text{softmax}(\varphi_\tau^{\text{prior}}(\mathbf{z}_{t\text{-}1})) \tag{2}$$

$$\begin{aligned} p(\mathbf{x}_{\leq T}, \mathbf{z}_{\leq T}) &= \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{<t}) p(\mathbf{z}_t|\mathbf{x}_{<t}, \mathbf{z}_{<t}) \\ &\approx \prod_{t=1}^{T} p(\mathbf{x}_t|\mathbf{z}_{\leq t}, \mathbf{x}_{<t}) p(\mathbf{z}_t|\mathbf{z}_{t\text{-}1}) \end{aligned} \tag{3}$$

Generation. $\mathbf{z}_t$ is a summarization of the current conversational exchange under the context. We use $\mathbf{z}_t$ and $\mathbf{h}_{t\text{-}1}$ to reconstruct the current utterance $\mathbf{x}_t$. This regeneration of $\mathbf{x}_t$ allows us to recover the dialog structure.

We use two RNN decoders, *dec1* and *dec2*, parameterized by γ_1 and γ_2 to generate the original u_t and s_t respectively. c_t and d_t are the hidden states of *dec1* and *dec2*. The context $\mathbf{h}_{\text{t-1}}$ and feature extraction vector $\varphi_\tau^{\mathbf{z}}(\mathbf{z}_t)$ are concatenated to form the initial hidden state h_0^{dec1} of *dec1*. $c_{(n_w,t)}$ is the last hidden state of *dec1*. Since v_t is the response of u_t and will be affected by u_t, we concatenate $c_{(n_w,t)}$ to d_0 to pass the information from u_t to v_t. This concatenated vector is used as h_0^{dec2} of *dec2*. This process is shown in Eq. (4) and (5).

$$\begin{aligned} &c_0 = [\mathbf{h}_{t-1}, \varphi_\tau^{\mathbf{z}}(\mathbf{z}_t)], \\ &w_{(i,t)}, c_{(i,t)} = f_{\gamma_1}(w_{(i-1,t)}, c_{(i-1,t)}) \end{aligned} \tag{4}$$

$$\begin{aligned} &d_0 = [\mathbf{h}_{t-1}, \varphi_\tau^{\mathbf{z}}(\mathbf{z}_t), c_{(n_w,t)}], \\ &v_{(i,t)}, d_{(i,t)} = f_{\gamma_2}(v_{(i-1,t)}, d_{(i-1,t)}) \end{aligned} \tag{5}$$

Recurrence. The state-level RNN updates its hidden state $\mathbf{h}_t$ with $\mathbf{h}_{\text{t-1}}$ based on the following Eq. (6). f_θ is a RNN parameterized by θ.

$$\mathbf{h}_t = f_\theta(\varphi_\tau^{\mathbf{z}}(\mathbf{z}_t), \varphi_\tau^{\mathbf{x}}(\mathbf{x}_t), \mathbf{h}_{t-1}) \tag{6}$$

Inference. We infer $\mathbf{z}_t$ from the context $\mathbf{h}_{\text{t-1}}$ and current utterances $\mathbf{x}_t$, and construct the posterior distribution of $\mathbf{z}_t$ by another softmax, shown in Eq. (7). Once we have the posterior distribution, we apply Gumbel-Softmax to take samples of $\mathbf{z}_t$. D-VRNN and DD-VRNN differ in their priors but not in their inference, because we assume the direct transitions between $\mathbf{z}_t$ in the prior instead of in the inference.

$$\mathbf{z}_t|\mathbf{x}_t \sim \text{softmax}([\varphi_\tau^{\text{enc}}(\mathbf{h}_{t-1}), \varphi_\tau^{\mathbf{x}}(\mathbf{x}_t)]) \tag{7}$$

Loss function. The objective function of VRNN is a timestep-wise variational lower bound, shown in Eq. (8) (Chung et al., 2015). To mitigate the *vanishing latent variable problem* in VAE, we incorporate bow-loss and Batch Prior Regularization (BPR) (Zhao et al., 2017, 2018) with tunable weights, λ to the final loss function, shown in Eq. (9).

$$\begin{aligned} &\mathcal{L}_{\text{VRNN}} = \mathbb{E}_{q(\mathbf{z}_{\le T}|\mathbf{x}_{\le T})}[\log p(\mathbf{x}_t \mid \mathbf{z}_{\le t}, \mathbf{x}_{<t})) + \\ &\sum_{t=1}^{T} \text{-KL}(q(\mathbf{z}_t \mid \mathbf{x}_{\le t}, \mathbf{z}_{<t}) \parallel p(\mathbf{z}_t \mid \mathbf{x}_{<t}, \mathbf{z}_{<t}))] \end{aligned} \tag{8}$$

$$\mathcal{L}_{\text{D-VRNN}} = \mathcal{L}_{\text{VRNN-BPR}} + \lambda * \mathcal{L}_{\text{bow}} \tag{9}$$

3.1 Transition Probability Calculation

A good way to represent a dialog structure both numerically and visually is to construct a transition probability table among latent states. Such transition probability can also be used to design reward function in the RL training process. We calculate transition table differently for D-VRNN and DD-VRNN due to their different priors.

D-VRNN. From Eq. (6), we know that $\mathbf{h}_t$ is a function of $\mathbf{x}_{\le t}$ and $\mathbf{z}_{\le t}$. Combining Eq. (1) and (6), we find that $\mathbf{z}_t$ is a function of $\mathbf{x}_{\le t}$ and $\mathbf{z}_{<t}$. Therefore, $\mathbf{z}_{<t}$ has an indirect influence on $\mathbf{z}_t$ through $\mathbf{h}_{\text{t-1}}$. This indirect influence reinforces our assumption that the previous states $\mathbf{z}_{<t}$ impacts future state $\mathbf{z}_t$, but also makes it hard to recover a clear structure and disentangle the direct impact of $\mathbf{z}_{\text{t-1}}$ on $\mathbf{z}_t$.

In order to better visualize the dialog structure and compare with the HMM-based models, we quantify the impact of $\mathbf{z}_{\text{t-1}}$ on $\mathbf{z}_t$ by estimating a bi-gram transition probability table, where $p_{i,j} = \frac{\#(\text{state}_i, \text{state}_j)}{\#(\text{state}_i)}$. The numerator is the total number of the ordered tuples ($\text{state}_{\text{i, t-1}}$, $\text{state}_{\text{j, t}}$) and the denominator is the total number of state_i in the dataset. We choose a bi-gram transition table over a n-gram transition table with a bigger n, as the most recent context is usually the most relevant, but it should be noted that unlike the HMM models, the degree of transition in our model is not limited nor pre-determined, because $\mathbf{z}_t$ captures all the context. Depending on different applications, different n may be selected.

DD-VRNN As stated before, the dependency of $\mathbf{z}_t$ on the entire context $\mathbf{h}_{\text{t-1}}$ creates difficulty in calculating the transition table. This is our motivation to derive the prior in DD-VRNN. The outputs from the softmax in the prior (Eq. (2)) directly constitute the transition table. So rather than estimating the transition probabilities by frequency count as in D-VRNN, we can optimize the loss function of DD-VRNN and get the parameters in Eq. (2) that directly form the transition table.

3.2 NE-D-VRNN

In task-oriented dialog systems, the presence of certain named entities, such as food preference plays a crucial role in determining the phase of the dialog. To make sure the latent states capture such useful information, we assign larger weights on the named entities when calculating the loss function in Eq. (9). The weights encourage the reconstructed utterances to have more correct named entities, therefore influencing the latent state to have better representation. We refer this model as NE-D-VRNN (Named Entitiy Discrete-VRNN).

4 Datasets

We test the proposed method on the CamRest676 corpus, which was released and collected by Wen et al. (2016). The task is to help users find restaurants in Cambridge, UK. While this task is highly similar to DSTC2, we choose this dataset instead of DSTC2 because it is relatively clean and comes with good entity extraction methods. There are a total of 676 dialogs in this dataset with three information slots (*food, price range and area*) and three request table slots (*address, phone and postcode*).

We also evaluate our model on another dataset of simulated conversations, proposed in Zhao and Eskenazi (2018). The task is to help users get the weather report in a certain place at a specific time. The dialog system is controlled by a fixed structure and hand-set probabilities. Therefore, learning the dialog structure of this dataset might be easier.

We assume each latent vector in the VAE emits one conversational exchange, including one user utterance and the corresponding system response at time t, and each conversational exchange corresponds to one latent vector, following Zhai and Williams (2014).

5 Experiments

We use LSTM (Hochreiter and Schmidhuber, 1997) with 200-400 units for the RNNs, and a fully-connected network for all the $\varphi_{\tau}^{(\cdot)}$ with a dropout rate of 0.4. Additionally, we use trainable 300-dimension word embeddings initialized by Google word2vec (Mikolov et al., 2013). The maximum utterance word length is 40 and the maximum dialog length is 10. We set the λ for the bow-loss to be 0.1. 80% of the entire dataset are used for training, 10% for validation and 10% for testing. Parameters mentioned are selected based on the performance of the validation set.

The evaluation of unsupervised methods has always been a challenge. We first compare our models with a simple K-means clustering algorithm to show its context sensitivity. Then we compare our models with traditional HMM methods both qualitatively and quantitatively. Finally, we compare the three proposed model variants. The qualitative evaluation involves generating dialog structures with different models, and the quantitative evaluations involves calculating the likelihood on a held-out test set under a specific model, which measures the model's predictive power.

5.1 Comparison with K-means Clustering

We apply the model and obtain a latent state $\mathbf{z}_t$ for each conversational exchange. Since $\mathbf{z}_t$ is a discrete one-hot vector, we can group the conversational exchanges with the same latent state together. This process is similar to clustering the conversational exchanges. But we choose the VAE over simple clustering methods because the VAE introduces more flexible context-sensitive information. A straightforward clustering method like K-means usually groups sentences with similar surface forms together, unless the previous context is explicitly encoded along with the utterances. To compare the grouping result of our model with a traditional clustering method, we perform K-means clustering on the dataset and calculate the within-cluster cosine similarity between the bag-of-word vectors of the utterances and the context. This cosine similarity measures how similar the utterances are on the word token level, higher the value is, more words the utterances share in common. It turns out the average cosine similarity between the current utterance is 0.536 using the K-means and 0.357 using the D-VRNN, while the average cosine similarity between the context is 0.320 using the K-means and 0.351 using the D-VRNN. This does show that in the D-VRNN result, the context are more similar to each other, while in the K-means, the current utterances are more similar on the surface textual level,which is not ideal because the dialog system needs context information. Table 1 shows an example where the D-VRNN clustering result is different from the K-means result. The conversational exchanges to be clustered are in the last row. These two exchanges have the same surface form but different contexts. D-VRNN identifies them as different by incorporating context information, whereas K-means places them into the same cluster ignoring the context.

5.2 Comparison with HMM

HMM is similar to our model. Actually, if we remove the $\mathbf{h}_t$ layer from Fig. 1, it becomes an HMM. But it is this additional layer of $\mathbf{h}_t$ that encodes the dialog context into continuous information and is crucial in the success of our model.

From	Utterance
SYS:	Okay, you don't care *place*, do you?
USR:	That's correct.
SYS:	What date are you interested?
USR:	**Weather this morning.**
SYS:	**I believe you said this morning.**

(a) One example in State 2, "provide place and time"

From	Utterance
USR:	Weather tomorrow.
SYS:	[api_call]. your weather report is here [report]. what else can I do?
USR:	**Weather this morning.**
SYS:	**I believe you said this morning.**

(b) One example in State 3, "additional time request"

Table 1: The utterances in bold in both tables have similar surface forms but different context. They are grouped in different latent states using the Discrete-VRNN. In contrast, they are in the same cluster using the K-means.

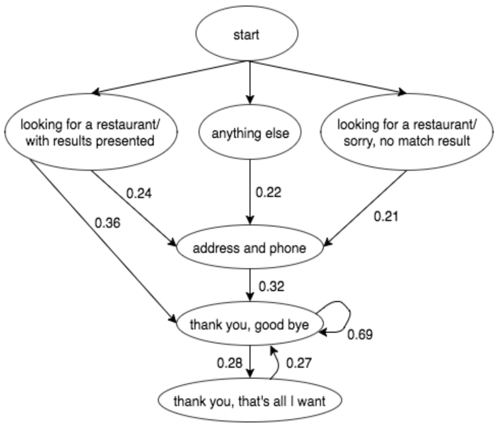

start
looking for a restaurant/ with results presented
anything else
looking for a restaurant/ sorry, no match result
0.24
0.22
0.21
0.36
address and phone
0.32
thank you, good bye
0.69
0.28
0.27
thank you, that's all I want

Figure 2: Dialog structure of restaurant data by D-VRNN. Transitions with $P \geq 0.2$ are visualized.

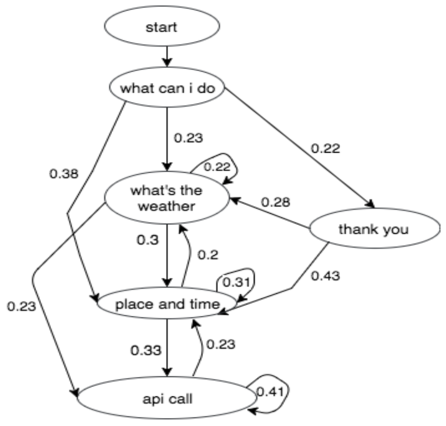


Figure 3: Dialog structure of weather data by D-VRNN. Transitions with $P \geq 0.2$ are visualized.

In Fig. 4 and 5, we compare our models quantitatively with the TM-HMM model with 10 topics and 20 topics from Zhai and Williams (2014), which performs the best on a similar task-oriented dialog dataset, the DSTC1 . The y-axis shows the negative log likelihood of reconstructing the test set under a specific model. The lower the negative log likelihood, the better the model performs. The x-axis shows different numbers of latent states N used in the models. As we can see, all of the VRNN-based models surpass the TM-HMM models by a large margin and are more invariant to the change in N on both datasets. Especially when N is small, the performance of HMM is not stable.

Qualitatively, we compare the dialog structures generated by different models. Fig. 2 and 3 show the discovered dialog structures using the D-VRNN model, and Fig. 7 in the Appendix shows the dialog structures learned by HMM. Each circle in these figures represents a latent state $\mathbf{z}_t$ with expert interpreted meanings, and the directed-arrows between the circles represent the transition probabilities between states. Human experts interpret each $\mathbf{z}_t$ consistently by going through conversational exchanges assigned to the same $\mathbf{z}_t$. For a better visualization effect, we only visualize the transitions with a probability equal or greater than 0.2 in the figures.

We observe reasonable dialog structures in Fig. 2 and 3. The D-VRNN captures the major path *looking for restaurant (anything else)* $\rightarrow$ *get restaurant address and phone* $\rightarrow$ *thank you* in the restaurant search task. It also captures *what's the weather* $\rightarrow$ *place and time* $\rightarrow$ *api call* in the weather report task. However, we do not get a dialog structure with entities separated (such as *food type I like* $\rightarrow$ *area I prefer* $\rightarrow$ *price range I want* $\rightarrow$ ...). Because users know the system's capability and tend to give as many entities as possible in a single utterance, so these entities are all mingled in one dialog exchange. But the model is able to distinguish the "presenting match result" state from the "presenting no match result" state (on the top of Fig. 2), which is important in making correct predictions. But in Fig. 7(a) generated by HMM, even if we set 10 states in the HMM, some states are still collapsed by the model because they share a similar surface form. And in Fig. 7(b) also by HMM, the dialog flow skips the "what can I do" state, and goes from the "start" directly to "providing place and time", which is not reasonable.

Another interesting phenomenon is that there are two "thank you concentrated" states in Fig. 2. This is because, users frequently say "thank you" on two occasions, 1) after the dialog system presents the restaurant information, most users will say "thank you"; 2) then the system will ask "is there anything else I can help you with", after which users typically respond with "thank you, that's all". This interesting structure is a reflection of the original rules in the dialog system. Moreover, in Fig. 3, we see 1) transitions from both directions between states "place and time" and "api call", and 2) transitions from "api call" to itself, as there are simulated speech recognition errors in the data and the system needs to confirm the entity values and update the API query.

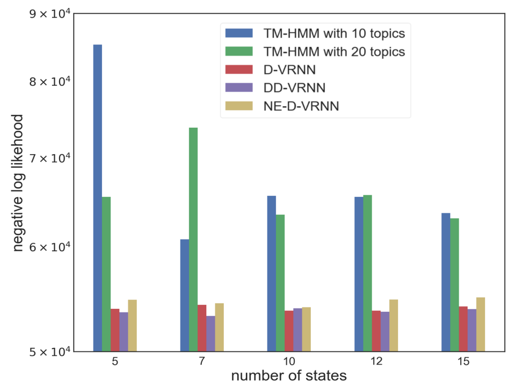


Figure 4: Negative log likelihood on restaurant data.

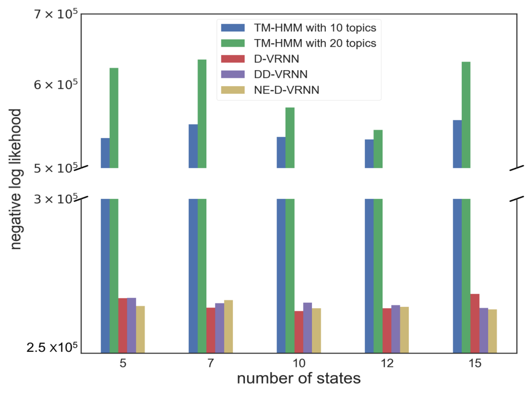


Figure 5: Negative log likelihood on weather data.

5.3 VRNN Model Variants Comparison

Even though the three proposed VRNN-based models have similar structures, they perform differently and are able to compensate each other. For example, in Fig. 8(b) in the Appendix, DD-VRNN is able to recognize a new state "not done yet, what's the weather", when users start a new query. This can complement D-VRNN's result.

Quantitatively, the three model variants also perform differently. On the restaurant test set shown in Fig. 4, DD-VRNN has the best overall performance compared with other models. Especially when the number of states N is small (e.g. 5 or 7 states), the advantage of the direct transition is more obvious. We think the reason behind is that it's easier and more accurate to model the direct transitions between a smaller set of states; as we increase N, the direct transitions between states become less and less obvious and therefore, help less on the predictive power. To our surprise, putting more weights on the named entities has a negative impact on the performance on the restaurant dataset. The underlying reason might be that the emphasis on the named entities shifts the focus of the model from abstract latent representation to a more concrete word token level. However, on the simulated weather dataset shown in Fig. 5, NE-D-VRNN performs relatively well. It might be because the weather dataset is completely simulated, which makes it easier to recognize the named entities. With a more accurate named entity recognition, NE-D-VRNN is able to help the performance. Overall, D-VRNN is the most stable one across datasets, so we will use D-VRNN in the following experiments.

6 Application on RL

The ultimate goal of such a structure discovery model is to utilize the structure to facilitate downstream tasks, such as dialog system training. Therefore, we propose to incorporate the dialog structure information into the reward function of the RL training. We believe that the transition table learned from the original dataset will guide the policy to make better decisions and converge faster by encouraging the policy to follow the real-data distribution. Similar to other RL models, we build a user simulator to perform the RL training. Please refer to the Appendix for the training details.

6.1 Reward Function Design

We use *policy gradient* method (Williams, 1992) to train the dialog policy. Traditional reward functions give a positive reward (e.g. 20) after the successful completion of the task, 0 or a negative reward to penalize a failed task and -1 at each extra turn to encourage the system to finish the task sooner rather than later (Williams et al., 2017). But this type of delayed reward functions doesn't have immediate rewards at each turn, which makes the model converge slowly. Therefore, we propose to incorporate the learned conversational exchange transition information as an immediate reward. The intuition is that in order to complete a task sooner, most users will follow a certain pattern when interacting with the system, for example, a typical flow is that users first give the entities information such as location, then ask for the restaurant information and finally, end the conversation. If we can provide the RL model with the information on what action is more likely to follow another action, the model can learn to follow real-data distributions and make better predictions. We encode the transition information through KL-divergence, a measurement of the distance between two distributions, in the reward function.

We design four types of reward functions and describe each of them in Algorithm 1 and Eq. 10 in the Appendix. The traditional delayed reward is the baseline. The second reward function, Rep-reward, uses constant penalty for repeated questions, as penalizing repetition yields better results, according to Shi and Yu (2018). The third reward function, KL-reward, incorporates the transition table information. From the RL model, we get the predicted probability p_{pred} for different actions; from the D-VRNN model, we get the transition probability p_{trans} between states and each state is translated to an action. We calculate the negation of the KL-divergence between p_{trans} and p_{pred} and use it as the immediate reward for every turn. This immediate reward links the predicted distribution with the real-data distribution by calculating the distance between them. The fourth reward function (KL+Rep) gives an additional -2 repetition penalty to the KL-reward to test the combination effect of the two types of penalties.

6.2 Result Analysis

We evaluate the RL performance by the average success rate, shown in Fig. 6. We observe that all the experimental reward functions greatly improve the performance of the baseline. We also observe that the baseline has a higher variance, which is due to the inefficient delayed rewards. Moreover, both the KL-reward and the KL-Rep reward reach a higher success rate at around 10,000 iterations than the Rep-reward and converges faster. These

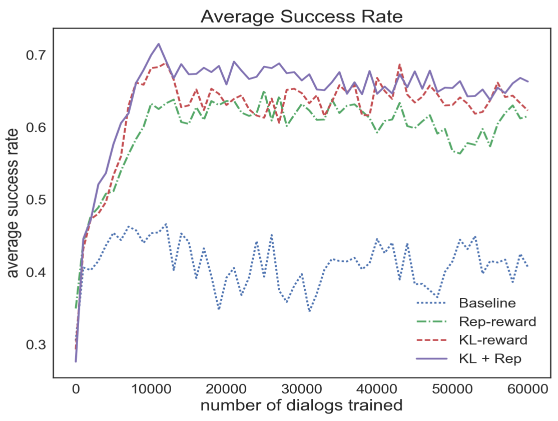


Figure 6: Average success rate of RL models with different reward functions.

two reward functions also achieve a better convergent success rate after 10,000 iterations. This suggests that adding KL-divergence of p_{trans} and p_{pred} into the reward helps develop a better policy faster.

The good performance comes from the transition probability p_{trans} learned by the discrete-VRNN. p_{trans} summarizes the communication pattern of most users in the real world and the KL-divergence measures the distance between the predicted distribution p_{pred} and p_{trans} from the real dataset. The RL model processes this KL-reward signal and learns to minimize the gap between p_{trans} and p_{pred}. As a result, p_{pred} will follow closer to the real data distribution, leading the model to converge faster to a better task success rate. In this way, we successfully incorporate the dialog structure information from the real data into the RL system training.

We observe that the use of KL reward improves the performance significantly in terms of both convergence rate and the final task success rate. Further, the combination of KL and repetition reward makes the model more stable and achieves a better task success rate, compared with the model with only KL-reward or Rep-reward. This indicates that the KL-reward can be combined with other type of rewards to achieve a better performance.

7 Conclusion and Future Work

A key challenge for discourse analysis and dialog system building is to extract the latent dialog structure. We adopted the VRNN with discrete latent variables to learn the latent states of each conversational exchange and the transitions between these states in an unsupervised fashion.

We applied the algorithm on a restaurant search task and a simulated weather report task, and evaluated the model quantitatively and qualitatively. We also proposed a way to incorporate the learned dialog structure information into a downstream dialog system building task. We involved the dialog structure in the RL reward design, which made the model converge faster to a better task success rate.

The performance of the Discrete-VRNN model has a major impact on the performance of the policy training. We plan to further improve the dialog structure learning process. Currently, we try to capture the status of the named entities by increasing the weights on the entities, which focus on the concrete word token level. In the future, we may use more sophisticated ways to encode the entity information into the latent states.

References

Ananlada Chotimongkol. 2008. *Learning the structure of task-oriented conversations from the corpus of in-domain dialogs*. Ph.D. thesis, Carnegie Mellon University, Language Technologies Institute, School of Computer Science.

Junyoung Chung, Kyle Kastner, Laurent Dinh, Kratarth Goel, Aaron C Courville, and Yoshua Bengio. 2015. A recurrent latent variable model for sequential data. In *Advances in neural information processing systems*, pages 2980–2988.

Carl Doersch. 2016. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*.

Emmanuel Ferreira and Fabrice Lefèvre. 2013. Expert-based reward shaping and exploration scheme for boosting policy learning of dialogue management. In *Automatic Speech Recognition and Understanding (ASRU), 2013 IEEE Workshop on*, pages 108–113. IEEE.

Barbara J Grosz and Candace L Sidner. 1986. Attention, intentions, and the structure of discourse. *Computational linguistics*, 12(3):175–204.

R Chulaka Gunasekara, David Nahamoo, Lazaros C Polymenakos, Jatin Ganhotra, and Kshitij P Fadnis. 2017. Quantized-dialog language model for goal-oriented conversational systems. *Dialog State Tracking Challenge 6, COLIPS*.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Dan Jurafsky. 1997. Switchboard swbd-damsl shallow-discourse-function annotation coders manual. *Institute of Cognitive Science Technical Report*.

Diederik P Kingma, Shakir Mohamed, Danilo Jimenez Rezende, and Max Welling. 2014. Semi-supervised learning with deep generative models. In *Advances in Neural Information Processing Systems*, pages 3581–3589.

Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Jingjing Liu, Stephanie Seneff, and Victor Zue. 2010. Dialogue-oriented review summary generation for spoken dialogue recommendation systems. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 64–72. Association for Computational Linguistics.

Yishu Miao and Phil Blunsom. 2016. Language as a latent variable: Discrete generative models for sentence compression. *arXiv preprint arXiv:1609.07317*.

Yishu Miao, Edward Grefenstette, and Phil Blunsom. 2017. Discovering discrete latent topics with neural variational inference. *arXiv preprint arXiv:1706.00359*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Gabriel Murray, Steve Renals, and Jean Carletta. 2005. Extractive summarization of meeting recordings.

Alan Ritter, Colin Cherry, and Bill Dolan. 2010. Unsupervised modeling of twitter conversations. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 172–180. Association for Computational Linguistics.

Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio. 2017. A hierarchical latent variable encoder-decoder model for generating dialogues. In *AAAI*, pages 3295–3301.

Weiyan Shi and Zhou Yu. 2018. Sentiment adaptive end-to-end dialog systems. *arXiv preprint arXiv:1804.10731*.

Michel Tokic. 2010. Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer.

Tsung-Hsien Wen, David Vandyke, Nikola Mrksic, Milica Gasic, Lina M Rojas-Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. 2016. A network-based end-to-end trainable task-oriented dialogue system. *arXiv preprint arXiv:1604.04562*.

Jason D Williams, Kavosh Asadi, and Geoffrey Zweig. 2017. Hybrid code networks: practical and efficient end-to-end dialog control with supervised and reinforcement learning. *arXiv preprint arXiv:1702.03274*.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Steve Young. 2006. Using pomdps for dialog management. In *Spoken Language Technology Workshop, 2006. IEEE*, pages 8–13. IEEE.

Ke Zhai and Jason D Williams. 2014. Discovering latent structure in task-oriented dialogues. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 36–46.

Tiancheng Zhao and Maxine Eskenazi. 2018. Zero-shot dialog generation with cross-domain latent actions. *arXiv preprint arXiv:1805.04803*.

Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi. 2018. Unsupervised discrete sentence representation learning for interpretable neural dialog generation. *arXiv preprint arXiv:1804.08069*.

Tiancheng Zhao, Ran Zhao, and Maxine Eskenazi. 2017. Learning discourse-level diversity for neural dialog models using conditional variational autoencoders. *arXiv preprint arXiv:1703.10960*.

8 Appendices

8.1 Reward Functions in RL

Algorithm 1 Reward function

```
if success then
    $R = 20$
else if failure then
    $R = -10$
else if repeated question then
    $R = f_{\text{reward}}(p_{\text{trans}}, p_{\text{pred}})$
else if each proceeding turn then
    $R = -1$
end if
```

$$f_{\text{reward}}(p_{\text{trans}}, p_{\text{pred}}) = \begin{cases} -1 & \text{Baseline} \\ -5 & \text{Rep-reward} \\ -\text{KL}(p_{\text{trans}}, p_{\text{pred}}) & \text{KL-reward} \\ -\text{KL}(p_{\text{trans}}, p_{\text{pred}}) - 2 & \text{KL+Rep} \end{cases} \quad (10)$$

8.2 RL Simulation Setting

The main purpose of the user simulator is to provide named entities. Therefore, it is built by sampling utterances from the real dataset according to the current status of named entities. Table 2 shows one simulated dialog in the reinforcement learning. We design seven system template actions in the simulated setting which fall under four categories, *ask for entity*, *present results*, *give restaurant information* and *closing statement*. But there are n latent states learned from the real data, and n is not necessarily equal to four. To solve this mismatch in dimensions between p_{pred} and p_{tran}, human experts further collapse similar latent states and reduce the number of latent states to four, which represents the four main categories in the actions, *ask for entity*, *present results*, *give restaurant information* and *closing statement*. For example, the two "thank you concentrated" states in Fig. 2 will be combined to the *closing statement* category. The probability of each category is the sum of the original probabilities whose actions are now under the same category.

From	Utterance
USR:	Could you please help me find an [value_price] restaurant in the [value_area] [slot_area]?
SYS:	Do you have a [slot_food] preference?
USR:	Let's try [value_food], please
SYS:	[value_name] is a good restaurant matching your request. Is there anything else I can help you with?
USR:	What other options are there?
SYS:	[value_name] is a good restaurant matching your request. Is there anything else I can help you with?
USR:	Great. what is their [slot_phone]?
SYS:	Here is the info...

Table 2: A simulated dialog.

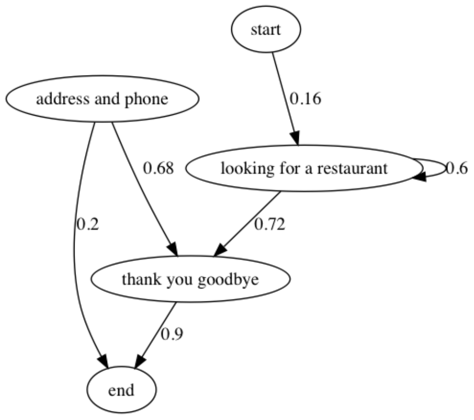
8.3 RL Training Details

A simple action mask is used to prevent impossible actions, such as presenting results without making a query to the DB. The input to the model is a list of common contextual features, such as the presence of each entity and the last action taken. The output of the model is the system action template.

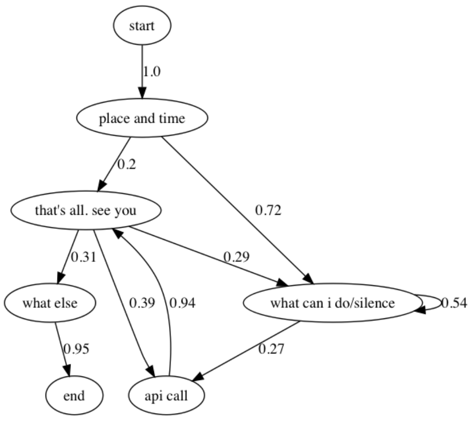
Four different reward functions are used. A discount factor of 0.9 is applied to all the experiments and the maximum number of turns is 10. An LSTM with 32 hidden units is used and the RL policy is updated after each dialog. We also apply ϵ-greedy exploration strategy (Tokic, 2010). Because the RL training can sometimes be unstable, we initialize all the model parameters using supervised learning on a simulated dataset, which consists of 500 dialogs between the user simulator and a rule-based agent simulator. The method is evaluated by freezing the policy after every 1000 updates and running 200 simulated dialogs to calculate the task success rate. We repeat the entire process 10 times and report the average success rate in Fig. 6.

8.4 Dialog Structures by Different Models

Fig. 7 and 8 below show the dialog structures generated by HMM and DD-VRNN.

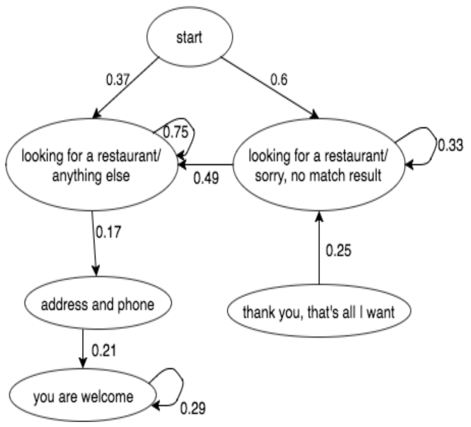


(a) HMM, restaurant data, 5 states

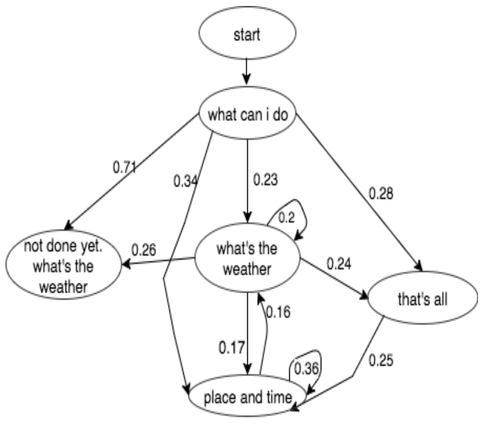


(b) HMM, weather data, 10 states

Figure 7: Dialog structures generated by HMM on different datasets. Transitions with $P \geq 0.2$ are visualized.


start
0.37
0.6
0.75
looking for a restaurant/ anything else
0.49
looking for a restaurant/ sorry, no match result
0.33
0.17
0.25
address and phone
thank you, that's all I want
0.21
you are welcome
0.29

(a) DD-VRNN, restaurant data, 5 states



(b) DD-VRNN, weather data, 10 states

Figure 8: Dialog structures generated by DD-VRNN on different datasets. Transitions with $P \geq 0.2$ are visualized.