24th European Conference on Artificial Intelligence - ECAI 2020
Santiago de Compostela, Spain

Unsupervised Learning of Interpretable Dialog Models

Dhiraj Madan [1] and **Dinesh Raghu** [2] and **Gaurav Pandey** [3] and **Sachindra Joshi** [4]

Abstract. Recently several deep learning based models have been proposed for end-to-end learning of dialogs. While these models can be trained from data without the need for any additional annotations, it is hard to interpret them. On the other hand, there exist traditional state based dialog systems, where the states of the dialog are discrete and hence easy to interpret. However these states need to be hand-crafted and annotated in the data. To achieve the best of both worlds, we propose Latent State Tracking Network (LSTN) using which we learn an interpretable model in unsupervised manner. The model defines a discrete latent variable at each turn of the conversation which can take a finite set of values. These variables correspond to the state of the dialog after each turn. Since the conversations are not labelled with the dialog states, we use EM algorithm to train our model in unsupervised manner. In the experiments, we show that LSTN can help achieve inter- pretability in dialog models with performance comparable to end-to-end approaches. This interpretability allows us to edit the model and improve the same.

1 Introduction

Recently, there have been several approaches [13, 4, 9, 5, 10] proposed for end-to-end learning of dialogs. Most of these approaches have an encoder-decoder architecture. The encoder understands the conversation so far by encoding it as a context vector, while the decoder generates the response based on the context vector. As the context vector is in continuous space, it is hard to interpret what the system has understood. Moreover, it is hard to interpret why a particular response was generated. More importantly, the model provides no means to control the type of responses the system can generate. In spite of being a black box, these approaches have gained popularity as they can easily adapt to new domain and do not require additional annotations on data.

On the other end of the spectrum are the traditional dialog systems. They cannot easily adapt to new domains as they require additional annotations on the data. However, they are interpretable and provide complete control over the system by restricting their state space to come from a discrete set. Here they have a set of *belief states* which represent what the system has understood so far. They also have a set of *action states* which represent the set of possible responses. At each turn of the conversation, the belief state is updated based on the user input and the previous belief state. The belief state is mapped to the action state, based on which a response is generated. These *state-based* dialog systems are usually designed as Markov decision processes or partially observable Markov decision processes. Unfortunately, human-intervention is necessary to define these states and annotate each dialog in the data that makes it hard to scale to new domains.

Recently, there has been a push towards reducing the amount of human intervention in state-based dialog systems [16] without compromising on interpretability. [16] proposed a deep learning based approach that learns the action space of a state-based dialog system in an unsupervised manner. However, the approach still requires the belief state to be hand-crafted and annotated for each turn in a dialog. There are also some efforts [18] for making end-to-end dialog systems more interpretable. Zhao et al. proposed a modification to end-to-end models where in they augment the context vector with a discrete valued vector resulting in interpretable responses. However, since the states are still in the continuous space, it is challenging to deduce the sequence of reasoning that led to a certain response, thereby making the transitions uninterpretable. Thus we see that the two strands of works are steadily moving towards the common goal of building a fully interpretable dialog model without the need for human intervention.

In this paper, we propose an approach for unsupervised learning of fully interpretable dialog models. We propose a Latent State Tracking Network (LSTN) to learn internal discrete states in an unsupervised manner. The network encodes the conversation-so-far into a discrete latent state using a transition model, while the emission model generates a response based on the encoded state. Since the proposed model is unsupervised, the discrete states are not available during training. Hence we propose an expectation-maximization (EM) based solution for jointly learning the states as well as the transition and emission models. Once the model has been trained, we can infer the state associated with a new user utterance using the transition model. Furthermore, we can generate the response that corresponds to the state using the emission model. The approach also allows us to construct a *dialog flow* which is a finite state automata with edges labelled with user utterances and states labelled with agent responses. This dialog flow can then be directly used to create bots using frameworks such as Google Dialog Flow [5], IBM Watson Assistant [6] and Microsoft Bot Framework[7]. One can traverse the dialog flow to generate novel conversations and edit the edges and vertices to further improve the dialog flow for conversation modelling.

To summarize, we make the following contributions: (1) We propose a novel Latent State Tracking Network (LSTN) for learning interpretable dialog models from conversations without any supervision. (2) We propose an EM-algorithm for jointly learning the latent states as well as the transition and emission modules in an LSTN. (3) We also show that in this process of discretization we do not lose much over the state-of-the-art, deep learning models for dialog, but gain in terms of having an interpretable model which can be easily

[1] IBM Research AI, email : dmadan07@in.ibm.com
[2] IBM Research AI, email : diraghu1@in.ibm.com
[3] IBM Research AI, email : gpandey1@in.ibm.com
[4] IBM Research AI, email : jsachind@in.ibm.com

[5] https://dialogflow.com/
[6] https://www.ibm.com/cloud/watson-assistant/
[7] https://dev.botframework.com/

modified using domain knowledge. In our experiments we show that editing the model leads to an improved performance as judged by human evaluation scores.

2 Latent State Tracking Network

Let a dialogue $D = \{(x_1, y_1), \ldots, (x_N, y_N)\}$ be represented as a sequence of utterances where x_i and y_i are the user utterance and agent response at the i^{th} turn. Given a set of such dialogues, we wish to learn a interpretable dialog model $\mathbb{M}$ which encodes the conversation so far using a discrete state variable and then samples a value from the state variable to generate an agent response y_i.

The state variable $z_i \in \{1, .., K\}$ at any turn i, is computed using the user utterance x_i at turn i, along with the previous turn's state variable z_{i-1}. We model this dependency using the *transition distribution* $p(z_i|z_{i-1}, x_i)$. The agent response y_i is then generated based on the state variable z_i. The relation between the response and the discrete state variable is modeled using an emission distribution $p(y_i|z_i)$. An illustration of the flow of dialog using our latent state tracking network is shown in Figure 1.

A graphical model representation of LSTN is given in Figure 2. The joint distribution of the agent responses $\mathbf{y} = (y_1, \ldots, y_N)$ and the belief states $\mathbf{z} = (z_1, \ldots, z_N)$ given the user utterances $\mathbf{x} = (x_1, \ldots, x_N)$ for a given conversation can be written as:

$$p(\mathbf{z}, \mathbf{y}|\mathbf{x}) = \prod_{i=1}^{N} p(z_i|z_{i-1}, x_i)p(y_i|z_i) \tag{1}$$

Note that there are two key distributions in this model:

1. The transition distribution which models the probability of moving to a new state z_i given the previous state z_{i-1} and current user utterance x_i.
2. The emission distribution which models the probability of generating response y_i given the current state z_i.

In order to completely define the model, we need to explain the computation of the above distributions from the utterances in a conversation.

2.1 The Transition Distribution:

Here, we need to model the probability of observing a new state z_i given the previous state z_{i-1} and the user utterance x_i. We use an LSTM network to embed the user utterance to a hidden state representation $h(x_i)$. For modeling the transition distribution, the states $\{1, \ldots, K\}$ are represented using continuous vectors $\{v_1, \ldots, v_K\}$. Hence, for the state z_{i-1}, we fetch the corresponding vector representation $v_{z_{i-1}}$. This vector is then concatenated with the hidden state representation of the utterance and then fed to a classifier with softmax outputs. The classifier outputs a probability distribution over the next states. Hence, the probability of the next state z_i given the previous state z_{i-1} and the user utterance x_i is given by

$$p(z_i|z_{i-1}, x_i) = \text{softmax}(W[h(x_i); v_{z_{i-1}}] + b)\,, \tag{2}$$

where W, b, the network h and the embeddings v_z are parameters that are learnt during training.

2.2 The Emission Distribution:

Given the current state z_i, this distribution models probability of all possible responses. To model this distribution, the states $\{1, \ldots, K\}$ are represented using continuous vectors $\{r_1, \ldots, r_K\}$. We feed the embedding of the current state to the decoder LSTM which outputs a sequence of distributions over the words. The probability of a response $y_i = (w_1, \ldots, w_M)$ conditioned on the state z_i is given by

$$p(y_i|z_i) = \prod_{j=1}^{M} p(w_j|w_1, \ldots, w_{j-1}, z_i) \tag{3}$$

Algorithm 1 Training Algorithm

procedure COMPUTECOST
  Input: Dialog Utterances $\{(x^{(i)}, y^{(i)})\}_{i=1}^{N_i}$
  Parameter Weights Θ, Posterior $q(z_i|z_{i-1}, \mathbf{x}, \mathbf{y})$
  Output: Log likelihood $L(\Theta)$
  Compute $f_N(\Theta, z_{N-1})$ using the transition and
  emission distribution as defined in (9)
  for $i \leftarrow N-1$ downto 1 **do**
    Compute $f_i(\Theta, z_{i-1})$ from $f_{i+1}(\Theta, z_i)$ using the
    transition and emission distribution as defined
    in (8).
  return $f_0(\Theta, z_0 = 0)$

2.3 Common representation of z versus different representation

As discussed previously, we have a vector representation of the latent variable Z , v_Z for obtaining the transition probabilities $p(Z_i|Z_{i-1}, x_i)$. We have another representation r_Z for generating the corresponding response. Both these vectors are trainable parameters. We have tried two different approaches , one using a common representation for r_Z and v_z and another with different representations. We found that using a common representation works better on all the datasets except SMD datasets with Schedule and Navigate domains.

2.4 Training the LSTN

In order to train the model, we need to maximize the marginal log-likelihood of the responses given the user utterances. Hence, we need to marginalize out the states $\mathbf{z} = (z_1, \ldots, z_N)$ from the model. The corresponding marginal log-likelihood for a single conversation is given by

$$\begin{aligned} L(\Theta) &= \ln(p(\mathbf{y}|\mathbf{x}, \Theta)) \\ &= \ln\left(\sum_{\mathbf{z}} \prod_{i} p(z_i|z_{i-1}, x_i; \Theta)p(y_i|z_i; \Theta)\right). \end{aligned}$$

We use EM algorithm to maximize the above objective function.

In the first step, also referred to as the E-step in literature, we compute the posterior distribution over all the states of a given conversation based on our current estimate of the parameters i.e. $q(\mathbf{z}) = p(\mathbf{z}|\mathbf{x}, \mathbf{y}; \theta_{old})$. In the M-step, we maximize the expectation of the joint log-likelihood with respect to the posterior obtained in the E-step i.e. $\max_\theta E_{q(z)}(\ln(p(z|x, y; \theta)))$. We discuss these steps in further detail below.

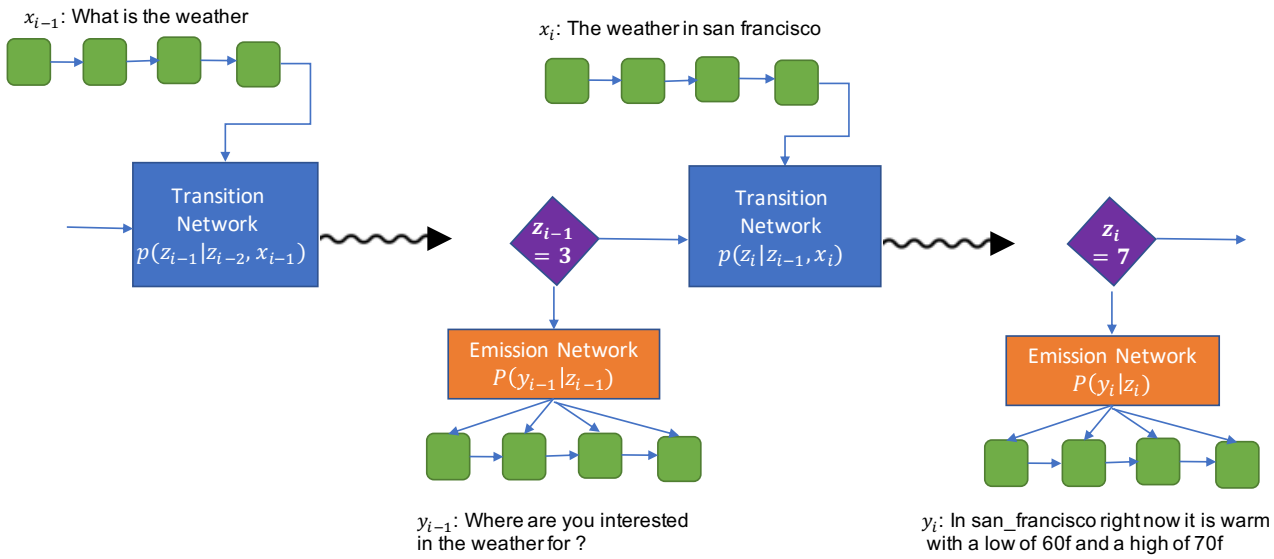

Figure 1. The Latent State Tracking Network for two steps of a conversation.

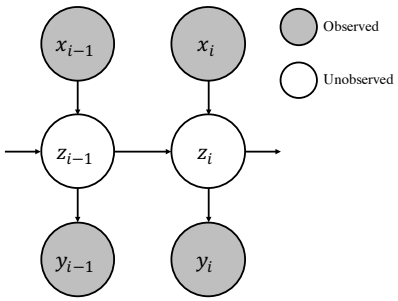

Figure 2. Plate notation of the Latent State Tracking Network

2.4.1 *The E-step:*

As discussed in the previous section, the prior distribution over the states of an LSTN given the user utterances factorizes as follows:

$$p(\mathbf{z}|\mathbf{x}) = \prod_{i=1}^{N} p(z_i|z_{i-1}, x_i), \tag{4}$$

where $z_0 = 0$ is the default state at the beginning of a conversation. Here, we will discuss the computation of the posterior distribution over the states given the user utterances and the agent responses. As with the prior, the posterior distribution over the states factorizes. That is,

$$p(\mathbf{z}|\mathbf{x}, \mathbf{y}) = \prod_{i=1}^{N} p(z_i|z_{i-1}, \mathbf{x}, \mathbf{y})$$

For the sake of brevity, we refer to $y_i, \ldots, y_N$ as $\mathbf{y}_{i:N}$. The same notation is used for sequence of user utterances and latent states. In order to compute the posterior, we note that given the previous state, the next state is independent of all previous agent responses. That is:

$$\begin{aligned} p(z_i|z_{i-1}, \mathbf{y}, \mathbf{x}) &= p(z_i|z_{i-1}, y_{i:n}, \mathbf{x}) \\ &\propto p(z_i, y_{i:n}|z_{i-1}, \mathbf{x}) \end{aligned}$$

To compute the above distribution, we use dynamic programming. In particular, the above distribution can be expressed in terms of the corresponding distribution at time step $i + 1$ as follows:

$$\begin{aligned} &p(z_i, y_{i:N}|z_{i-1}, \mathbf{x}) \\ &= p(z_i|z_{i-1}, x_i)p(y_i|z_i) \sum_{z_{i+1}} p(z_{i+1}, y_{i+1:N}|z_i, \mathbf{x}) \end{aligned} \tag{5}$$

Note that the distribution within the summation has the same form as the distribution that we wish to compute. Hence, the desired distribution at timestep i can be computed recursively from the corresponding distribution at timestep $i + 1$. Moreover, the distribution at the last timestep can be computed directly as follows:

$$p(z_N, y_N|z_{N-1}, \mathbf{x}) = p(z_N|z_{N-1}, x_N)p(y_N|z_N)$$

Thus, we can run the above computation over the N turns of the conversation to obtain the posterior distribution of each latent state.

2.4.2 *The M-step:*

Having obtained the posterior, we use it for maximizing the expected complete log-likelihood of the agent responses and the latent states. In particular, we need to maximize

$$\mathbb{E}_{\mathbf{z}\sim p(\mathbf{z}|\mathbf{y},\mathbf{x},\Theta^{old})} \ln p(\mathbf{z}, \mathbf{y}|\mathbf{x}, \Theta) \tag{6}$$

Here, Θ_{old} in the posterior refers to the fact that the posterior has been evaluated using the current parameters and will be held fixed during the M-step. The above expectation is computed recursively

using a Viterbi based approach. We define $f_i(\Theta, z_{i-1})$ as the expectation of the log-likelihood of the last $N - i$ states and agent responses. That is:

$$f_i(\Theta, z_{i-1}) = \mathbb{E}_{\mathbf{z}_{i:N}} \ln p(\mathbf{z}_{i:N}, \mathbf{y}_{i:N} | \mathbf{x}_{i:N}, z_{i-1}, \Theta) \,, \quad (7)$$

where the expectation is over the posterior distribution of the latent states. Note that the objective that we wish to optimize is $f_1(\Theta, z_0)$, where z_0 is the default start state. To compute this quantity, we note that $f_i(\Theta, z_{i-1})$ can be expressed as function of $f_{i+1}(\Theta, z_i)$ as follows:

$$\begin{aligned} f_i(\Theta, z_{i-1}) = \mathbb{E}_{z_i} [f_{i+1}(\Theta, z_i) \\ + \ln(p(z_i|z_{i-1}, x_i, \Theta)) + \ln(p(y_i|z_i, \Theta))] \,, \end{aligned} \quad (8)$$

where the expectation is over the posterior distribution of Z_i. Finally, we note that $f_N(\Theta, z_{n-1})$ can be computed directly to begin the recursion as follows:

$$\begin{aligned} f_N(\Theta, z_{N-1}) = &\mathbb{E}_{Z_N} [\ln(p(z_N|z_{N-1}, x_N, \Theta)) \\ &+ \ln(p(y_N|z_N, \Theta))] \end{aligned} \quad (9)$$

The computation of $f_1(\Theta, z_0)$ from $f_N(\Theta, z_N)$ constitutes the forward pass of the M-step and is listed in Algorithm 1. Note that each step of the computation is differentiable, and hence, the objective is a differentiable function of the transition and emission distributions. Hence, during the backward pass, we backpropagate the gradient all the way from the final objective to the transition and emission distributions.

2.5 Inference

In this section we will discuss how the trained model is used for generating the response utterance given the context consisting of previous user utterances and agent responses . There are two parts to our inference:-

2.5.1 Emission Module:

Here given a dialog state, we need to generate the mostly likely responses associated with the same. Having learnt the distribution $p(y|z)$, as a decoder RNN, we use this to generate top responses for each value of z. For each value of z from 1 to K, we initialize the hidden state of decoder RNN with vector r_z and perform beam search to generate the top responses. In our experiments we used a beam size of 10. This step is performed only once and is not repeated for new test examples. At test time it will suffice to use the top responses associated with a state or sample one from the top 10 generated through beam search.

2.5.2 Transition Module:

This module computes a distribution over current state given the past state and the new user utterance. During inference, we use this module to obtain the distribution of each state given the past user utterances. In particular, the distribution of the i^{th} state given all the user utterances till step i can be expressed as follows:
$p(z_i|\mathbf{x}_{1:i}) = \sum_{z_{i-1}} p(z_i|z_{i-1}, \mathbf{x}_i) p(z_{i-1}|\mathbf{x}_{1:i-1})$. During inference, we can generate the response corresponding to the most probable hidden state. i.e. we compute $\bar{z}_i = \arg\max_{z_i} p(z_i|\mathbf{x}_{1:i})$. We then produce the most likely response corresponding to $\bar{z}_i$ using emission module.

Algorithm 2 Inference Algorithm

```
procedure StateTracker
    Initialize state distribution as p(z_0 = 0) = 1 and i ← 1.
    while True do
        Receive new user utterance x_i
        Update p(z_i) = Σ_{z_{i-1}} p(z_{i-1}) p(z_i | z_{i-1}, x_i)
        Compute z̄_i = arg max_{z_i} p(z_i)
        Generate top response corresponding to z̄_i
        i ← i + 1
```

3 Experiments

3.1 Datasets

We perform experiments on Stanford Multi-Domain Dataset (SMD) [1], and Car Assistant Dialog Dataset (CADD). The SMD dataset contains conversations from three domains: calendar, navigation and weather. CADD is a set of 986 conversations between an in-house car assistant bot and its users. The bot is designed to help with navigation and controlling various devices in the car. SMD was pre-processed as in [1] to reduce lexical variability.

3.2 Training

Adam optimizer was used for training . The hyperparameters were selected based on perplexity on a held-out validation set. The learning rate was varied over the set $\{0.01, 0.001, 0.0001\}$, dimension of word embeddings from $\{16, 32, 64\}$, the number of distinct latent states K from $\{8, 16, 32, 64, 128\}$. We experimented with having same versus different embeddings for the latent states while computing transition and emission distributions.

Intent Class	Train User Utterances
#turn_on_music	music play music turn on the radio
#rock	i like rock oh rock then i prefer the genre of rock
#jazz	jazz play some jazz i hate jazz
#find_gas_station	find a gas station locate a gas station i need gas
#first	first 1 second please

Table 1. Intent classes and the associated user utterances from the CADD dataset

3.3 Interpretable Model

The discrete nature of LSTN enables it to be translated to a finite state automaton, whose transitions correspond to user utterances. To obtain this automaton, we run the inference algorithm of LSTN for each conversation in the training data. Each turn in a conversation is

labelled with a latent state z in this step. To obtain the transitions for each pair of states z and z', we identify the user utterance u in the conversation that led the transition from z to z'. The directed edge between the two nodes is labelled with the utterance x.

When we simulate the above procedure over all the conversations in the training data, we may obtain several user utterances that lead the transition between z and z'. Each such group of user utterances between a fixed pair of values z and z' correspond to an *intent class*. Few of the intent classes in the CADD dataset are listed in Table 1. As can be observed from the table, LSTN modeling causes similar user utterances to cluster together in the same intent class.

We further associated each node in the finite state automaton with an agent response. To obtain the response for a latent state z, we maximize the emission module $p(y|z)$ with respect to the response y. In practice, this can be achieved by running beam search on the emission module $p(r|z)$, and picking the response that maximizes this probability.

Having completed the above steps, we obtain a finite state automaton whose edges are labelled with user intents, while the vertices are labelled with agent responses. We refer to this automaton as dialog flow. This dialog flow can be directly consumed by chatbot building frameworks such as Google Dialog Flow , IBM Watson Assistant and Microsoft Bot Framework to create a chatbot. As the dialog flow is completely interpretable, a domain expert can edit the transitions and emissions to fine tune the bot as required.

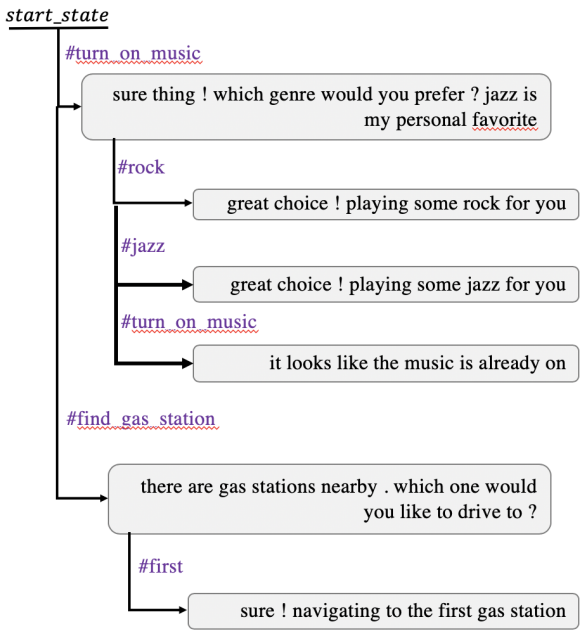

Figure 3. A part of the dialog flow for CADD dataset

A conversation can be generated by traversing the dialog flow and listing the user intents and agent responses of the nodes and edges visited, sequentially. A subtree of the dialog flow for CADD dataset is shown in Figure 3.

3.4 Baselines

To illustrate the advantage of jointly modeling the transitions and emissions (as in LSTN), we compare it against a model that learns the emissions and the transitions in a pipelined fashion. We call the latter model as *split-LSTN*. This model is learnt in two phases. In the first phase, the conversation-so-far is encoded using an LSTM and mapped to a discrete-value from which the response is generated. The likelihood of the response is maximized using EM algorithm to learn the latent states and the emissions. In the second phase, we learn the transitions between the latent states obtained in the previous phase in a supervised manner.

We further compare our model against a sequence-to-sequence model often employed for conversation modelling, namely, hierarchical recurrent encoder decoder (HRED) [11]. HRED translates a sequence of user utterances and agent responses to a continuous valued context vector. The agent response is generated from the context vector using a decoder.

3.5 Editing the LSTN

HRED is not an interpretable model, and hence, it is not possible to edit the model to improve the responses. In contrast, the proposed LSTN model can be edited by first converting the model into a dialog flow as described in the previous section and then editing the edges and vertices of the automaton. We refer to the edited model as *edited-LSTN*. In particular, we follow a two step process to edit dialog flows: 1) Split almost similar response clusters and 2) Merge examples from similar transitions together.

In some cases, two very similar responses get associated with the same state, say z. For example, in the CADD dataset, "i 'll turn off the lights for you" and "i 'll turn off the wipers for you" are associated with the same state. The former response has the highest probability of being generated given the state z. So, during test time it would be picked when the user requests to turn off either the lights or the wipers. The first step helps to alleviate this issue by spliting the state into two (i.e) a new state z' would now be created with a response "i 'll turn off the wipers for you". Transitions to/from the new state z' are then created by splitting the transitions which include z.

Current user utterance and the previous discrete state decide the current discrete state. In other words, user utterances that helps move the state from a state z to another state z' are clustered together. In some cases, the current state is dependent only on the current user input but not the previous state. For example, when the user input is "Thank you", irrespective of what the previous state is, the reply is "you are welcome". In such cases, forcing the transition to be conditioned on the previous state, creates sparsity. In some cases, more that one transition to a state z would semantically similar. So, as a second step, we identify such transitions and merge the examples together to reduce sparsity.

While the dialog flow discussed in Section 3.3 was obtained in a completely automated manner, we involve a human-in-the-loop to edit the dialog flow. However, the effort involved in editing the dialog flow is much smaller than building an entire dialog flow from scratch.

3.6 Quantitative Evaluation

Evaluating dialogue models is a significantly challenging problem on its own. One can use perplexity to evaluate how well a given probabilistic model fits the data. However, when the data contains a lot of generic utterances, a model can achieve low perplexity by assigning

Dataset	BLEU Score				Human Evaluation		
	HRED	Split-LSTN	LSTN	Edited-LSTN	HRED	LSTN	Edited-LSTN
SMD (Wea.)	15.59	13.41	**16.16**	15.92	0.84	0.90	**0.92**
SMD (Cal.)	**17.90**	14.78	15.90	17.30	0.80	0.80	**0.88**
SMD (Nav.)	**9.09**	7.51	8.03	7.98	0.78	0.68	**0.84**
CADD	**63.88**	56.05	57.98	49.65	0.54	0.62	**0.86**

Table 2. Comparison of BLEU Scores of responses generated from HRED, split-LSTN, LSTN and edited-LSTN Connections

high probability to generic utterances. In contrast, a model that generates non-generic utterances specific to the conversation could still have high perplexity.

Alternatively, one can employ n-gram based measures such as BLEU to determine the overlap between the generated and ground-truth response. Unlike perplexity, BLEU score can be used for comparing non-probabilistic models. While BLEU scores are often used for comparing models in translation and speech recognition, they are quite unsuitable for comparing dialog models since two responses with no n-gram overlap could be equally suitable for a given context.

Finally, since the BLEU scores can be uncorrelated with human judgements for dialogs [6], we also performed human annotations on all the 4 datasets. In particular, we randomly selected 50 context-response pairs for each dataset. We generated the responses using 3 models: HRED, LSTN and Edited-LSTN to obtain a total of 600 annotations. For each context, the annotator was asked to assign a binary value for each annotated response, where a 1 indicates that the response is suitable for the current context, while a 0 indicates its unsuitability.

Table 2 lists the BLEU scores as well as the scores for human evaluation obtained on various datasets. As can be observed from the Table, non-interpretable sequence-to-sequence models achieve higher BLEU score than interpretable models. Among interpretable models, LSTNs achieve better BLEU score than split-LSTN, thereby suggesting that jointly optimizing the transition and emission modules is beneficial for conversation modelling.

While LSTN and Edited-LSTN achieve lower BLEU scores, they perform better on human evaluations. This is due to the fact that a single context can be associated with multiple responses. For instance, as shown in Table 3, the responses generated by HRED as well as edited-LSTN are suitable for the context. However, the responses generated by Edited-LSTN will have much lower BLEU score than the one generated by HRED. It is also worthy to note that editing the LSTN model often results in a decrease in BLEU scores. This is expected since we do not care about maximizing quantitative metrics while editing LSTN. To evaluate the effect of the number of latent states K, we trained LSTN for several values for K. Figure 4 shows the performance in terms of BLEU score for two datasets. We observe that as the value of K increases the performance also increases and saturates after a while.

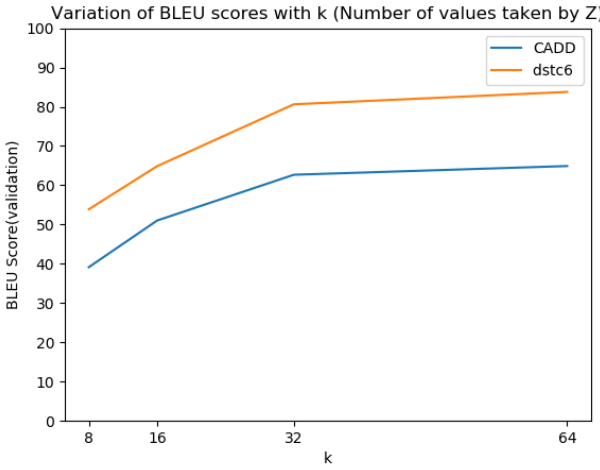

Figure 4. Variation of BLEU Score with number of latent states K

4 Related Work

To the best of our knowledge, our work is the first to propose an unsupervised approach for learning interpretable dialog models. The word unsupervised indicates that the dialogs used to train the model are not annotated with any labels.

Unsupervised Learning of Dialogs: Early approaches for learning dialogs from chat transcripts were inspired from machine translation [8] and language modelling [11]. [13] ([13]) proposed a deep learning approach based on the sequence-to-sequence model [12]. [9] ([9]) proposed an approach that leverages the hierarchical structure of the dialog to model them better. Even though these models can be trained in an unsupervised manner, they are not interpretable.

Interpretable Dialog Models Traditional task oriented dialog systems were built using reinforcement learning (RL) approaches. The systems were either modeled as Markov decision processes [3, 14] or partially observable Markov decision processes [17, 2]. Recently, there have been efforts [15, 16] to solve RL based approaches using deep learning. The discrete states and actions make the model easy to interpret. However this comes at a cost of handcrafting states, actions and rewards for training the RL model. In our approach, we provide interpretability by learning discrete intermediate states latently without any handcrafting.

Discrete Latent Variables: An emerging area in deep learning research is to use a set of discrete latent variables in the deep learning model. Learning dialogs by augmenting the real-valued context vector with discrete latent variables [16] adds a notion of interpretability. The prior works only use the discrete state to generate the response. However the transitions are still in continuous space. Another work [7] also considers the idea of finding dominant clusters in the user utterances and the response space. However this work only considers clusters based on utterance level. Also the transitions between the states are not learnt. Our work differs from these approaches as we use discrete latent states to capture the entire conversation context.

User	hi there
Gold Response	hello ! i 'm doing good . i 'm here to help you . just say the word
HRED	hello ! i 'm doing good . i 'm here to help you . just say the word
Edited-LSTN	hi . it looks like a nice drive today . what would you like me to do ?
User	turn on wipers
Gold Response	hi . it looks like a nice drive today . what would you like me to do ?
HRED	i 'll turn on the lights for you
Edited-LSTN	i 'll turn on the wipers for you
User	thank you .
Gold Response	you 're welcome !
HRED	you 're welcome !
Edited-LSTN	no problem .

Table 3. Examples of responses generated by HRED and edited-LSTN

The discrete states and the transitions between them help us in making the entire model easy to interpret and modify. This also allows us to create a dialog flow as discussed in Section 3.3 .

5 Conclusions

In this paper, we introduce a novel problem of learning interpretable dialog models in an unsupervised manner. We propose a novel model, Latent State Tracking Network (LSTN) for this task. LSTN learns the discrete latent states using a EM based algorithm. We show that (1) even after discretization the states learnt by LSTN are as good as uninterpretable models (such as HRED) (2) joint learning of emissions and transitions is better than learning them in a pipelined manner and (3) the learnt emissions and transitions are interpretable and meaningful.

REFERENCES

[1] Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning, 'Key-value retrieval networks for task-oriented dialogue', in *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue, Saarbrücken, Germany, August 15-17, 2017*, pp. 37–49, (2017).

[2] Milica Gasic, Catherine Breslin, Matthew Henderson, Dongho Kim, Martin Szummer, Blaise Thomson, Pirros Tsiakoulis, and Steve Young, 'Pomdp-based dialogue manager adaptation to extended domains', in *Proceedings of the SIGDIAL 2013 Conference*, pp. 214–222, (2013).

[3] Esther Levin, Roberto Pieraccini, and Wieland Eckert, 'A stochastic model of human-machine interaction for learning dialog strategies', *IEEE Transactions on Speech and Audio Processing*, **8**(1), 11–23, (2000).

[4] Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan, 'A diversity-promoting objective function for neural conversation models', in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 110–119, (2015).

[5] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky, 'Deep reinforcement learning for dialogue generation', in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 1192–1202, (2016).

[6] Chia-Wei Liu, Ryan Lowe, Iulian Serban, Mike Noseworthy, Laurent Charlin, and Joelle Pineau, 'How not to evaluate your dialogue system: An empirical study of unsupervised evaluation metrics for dialogue response generation', in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pp. 2122–2132, (2016).

[7] Dhiraj Madan and Sachindra Joshi, 'Finding dominant user utterances and system responses in conversations', *arXiv preprint arXiv:1710.10609*, (2017).

[8] Alan Ritter, Colin Cherry, and William B Dolan, 'Data-driven response generation in social media', in *Proceedings of the conference on empirical methods in natural language processing*, pp. 583–593. Association for Computational Linguistics, (2011).

[9] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau, 'Building end-to-end dialogue systems using generative hierarchical neural network models.', in *AAAI*, volume 16, pp. 3776–3784, (2016).

[10] Iulian Vlad Serban, Alessandro Sordoni, Ryan Lowe, Laurent Charlin, Joelle Pineau, Aaron C Courville, and Yoshua Bengio, 'A hierarchical latent variable encoder-decoder model for generating dialogues.', in *AAAI*, pp. 3295–3301, (2017).

[11] Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan, 'A neural network approach to context-sensitive generation of conversational responses', in *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 196–205, (2015).

[12] Ilya Sutskever, Oriol Vinyals, and Quoc V Le, 'Sequence to sequence learning with neural networks', in *Advances in neural information processing systems*, pp. 3104–3112, (2014).

[13] Oriol Vinyals and Quoc Le, 'A neural conversational model', *Proceedings of the International Conference on Machine Learning, Deep Learning Workshop.*, (2015).

[14] Marilyn Walker, Rashmi Prasad, and Amanda Stent, 'A trainable generator for recommendations in multimodal dialog', in *Eighth European Conference on Speech Communication and Technology*, (2003).

[15] TH Wen, D Vandyke, N Mrkšíc, M Gašíc, LM Rojas-Barahona, PH Su, S Ultes, and S Young, 'A network-based end-to-end trainable task-oriented dialogue system', in *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017-Proceedings of Conference*, volume 1, pp. 438–449, (2017).

[16] Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young, 'Latent intention dialogue models', in *International Conference on Machine Learning*, pp. 3732–3741, (2017).

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

[18] Tiancheng Zhao, Kyusong Lee, and Maxine Eskenazi, 'Unsupervised discrete sentence representation learning for interpretable neural dialog generation', in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1098–1107. Association for Computational Linguistics, (2018).