Dialogue State Induction Using Neural Latent Variable Models

Qingkai Min[1,2] , **Libo Qin**[3] , **Zhiyang Teng**[1,2] , **Xiao Liu**[4] and **Yue Zhang**[1,2]
[1]School of Engineering, Westlake University
[2]Institute of Advanced Technology, Westlake Institute for Advanced Study
[3]Research Center for Social Computing and Information Retrieval, Harbin Institute of Technology
[4]School of Computer Science and Technology, Beijing Institute of Technology
{minqingkai,tengzhiyang,zhangyue}@westlake.edu.cn, lbqin@ir.hit.edu.cn, xiaoliu@bit.edu.cn

Abstract

Dialogue state modules are a useful component in a task-oriented dialogue system. Traditional methods find dialogue states by manually labeling training corpora, upon which neural models are trained. However, the labeling process can be costly, slow, error-prone, and more importantly, cannot cover the vast range of domains in real-world dialogues for customer service. We propose the task of dialogue state induction, building two neural latent variable models that mine dialogue states automatically from unlabeled customer service dialogue records. Results show that the models can effectively find meaningful dialogue states. In addition, equipped with induced dialogue states, a state-of-the-art dialogue system gives better performance compared with not using a dialogue state module.

1 Introduction

Dialogue state modules are a central component to a task-oriented dialogue system [Wen *et al.*, 2017; Lei *et al.*, 2018]. Given a user utterance and existing dialogue history, a dialogue system typically extracts dialogue states, according to which a system response is generated. An example is shown in Figure 1, given two turns of a dialogue, the first user utterance is "I want an expensive restaurant that serves Turkish food.", and the dialogue states consist of the slot-value pairs `inform(price=expensive, food=Turkish)`. As the dialogue proceeds, the dialogue state is updated at each turn. After tow dialogue turns, the dialogue state becomes `inform(price=expensive, food=Turkish); request(area)`, where *inform* represents the search constraints expressed by user and *request* represents the search target that the user is asking for. In this example, the user intention is to reserve a restaurant. The business domain is restaurant customer service. The dialogue state represents what the user is looking for at the current turn of the conversation.

Prior work has mostly followed a manual labeling-train-test paradigm, which begins with the design of annotation guidelines, followed by the collection and manual labeling of training corpora, before training a model. The supervised learning task is called *Dialogue State Tracking* (DST) [Young

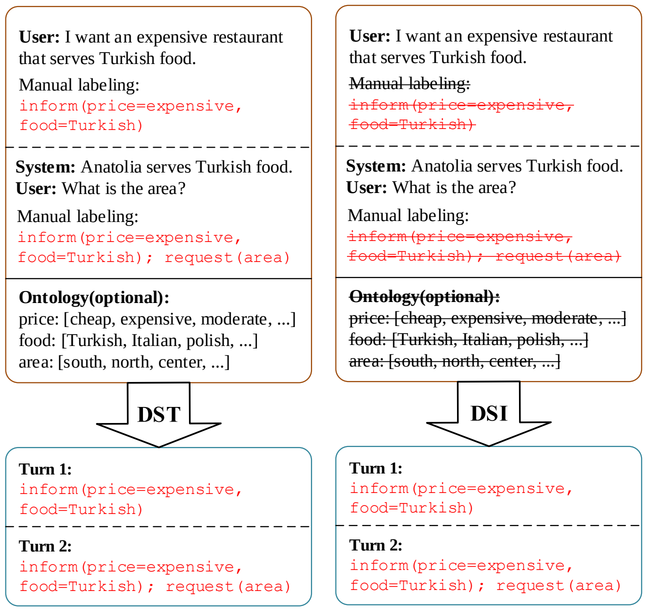

User: I want an expensive restaurant that serves Turkish food.
Manual labeling:
inform(price=expensive, food=Turkish)
System: Anatolia serves Turkish food.
User: What is the area?
Manual labeling:
inform(price=expensive, food=Turkish); request(area)
Ontology(optional):
price: [cheap, expensive, moderate, ...]
food: [Turkish, Italian, polish, ...]
area: [south, north, center, ...]
User: I want an expensive restaurant that serves Turkish food.
~~Manual labeling:~~
~~inform(price=expensive, food=Turkish)~~
System: Anatolia serves Turkish food.
User: What is the area?
~~Manual labeling:~~
~~inform(price=expensive, food=Turkish); request(area)~~
~~Ontology(optional):~~
~~price: [cheap, expensive, moderate, ...]~~
~~food: [Turkish, Italian, polish, ...]~~
~~area: [south, north, center, ...]~~
DST
DSI
Turn 1:
inform(price=expensive, food=Turkish)
Turn 2:
inform(price=expensive, food=Turkish); request(area)
Turn 1:
inform(price=expensive, food=Turkish)
Turn 2:
inform(price=expensive, food=Turkish); request(area)

Figure 1: Comparison between DSI and traditional DST. The strikethrough font is used to represent the resources not needed by DSI. The dialogue state is accumulated as the dialogue proceeds. Turns are separated by dashed lines. Dialogues and external ontology are separated by black lines.

et al., 2010]. One limitation for supervised learning is that the manual labeling process can be slow and costly given a certain domain of customer service. Available datasets are labeled on a few popular domains such as restaurant, taxi, train and hotel [Williams *et al.*, 2014; Budzianowski *et al.*, 2018; Rastogi *et al.*, 2019]. However, in practice, the number of customer service domains ranges far beyond hundreds (e.g. telecommunication customer service, banking, household maintenance, police, e-commercial customer service, etc), which makes it infeasible to manually label corpora for every domain. In addition, it has been shown that the ratio of annotation errors can be as high as 30% and even 40% for the DST task (i.e., the MultiWOZ datasets) [Eric *et al.*, 2019; Zhang *et al.*, 2019].

To address this issue, it can be useful to automatically induce dialogue states from raw dialogues. We assume that there is a large set of dialogue records of many different domains, but without manual labeling of dialogue states. Such data are relatively easy to obtain, for example from customer service call records from different businesses. Consequently, we propose the task of **dialogue state induction** (DSI), which is to automatically induce slot-value pairs from raw dialogue data and can be better used for downstream dialogue tasks such as database query, act prediction and response generation. The difference between DSI and DST is illustrated in Figure 1. Similar to DST models, DSI outputs dialogue states in slot-value pairs such as `inform(price=expensive)`, where *price* represents a slot, and *expensive* represents a value. For requestable slots such as `request(area)`, *request* is regarded as the slot and *area* is regarded as the value. During training, DST relies on both a dialogue record and manual labeling of slot-value pairs on the dialogues. In contrast, our task does not rely on manual labeling and can generate slot-value pairs over raw dialogues automatically.

We introduce two neural latent variable models for DSI by treating the whole state and each slot as latent variables, from which values observed in dialogue data are generated. The goal is to induce slots according to those frequently co-occurring values and the dialogue contexts. In particular, each value (i.e., phrase in raw text) is represented by using both a sparse one-hot representation and a dense contextualized embedding representation. Both models are generative probabilistic models, which generate a value by first generating a latent dialogue state vector, and then generating a slot. The difference between the two models is the modeling of service domains. We observe that different service domains may contain slots with similar contexts or values. For example, both *taxi* and *bus* domains can have the same slot *to_location*. In order not to mix their structures from a large dialogue record, our second model further considers the service domain explicitly by taking the dialogue state as a *Mixture-of-Gaussians*. We refer to the basic model *DSI-base* and the advanced model *DSI-GM*.

Experiments over the MultiWOZ [Budzianowski *et al.*, 2018] and the SGD [Rastogi *et al.*, 2019] datasets show that both DSI models can effectively induce dialogue states compared with a random select strategy. In addition, the Gaussian mixture model gives significantly better results compared with the basic model. Finally, we apply DSI to a recently proposed dialogue system [Chen *et al.*, 2019], by replacing the dialogue state module with our *DSI-GM* model. Results show that adding induced dialogue states gives significantly better results in both dialogue act prediction and response BLEU compared with a dialogue system without considering dialogue states. In particular, the BLEU score using the *DSI-GM* model outputs is better by 2.1% compared with not using dialogue states, and lower by 0.8% compared with using manual labeling dialogue states. This shows that by inducing dialogue states, improved dialogue systems can be obtained. To our knowledge, we are the first to automatically induce dialogue states in the form of slot-value pairs using a neural latent variable model. Our models can serve as baselines for further research. We release our code at https://github.com/taolusi/dialogue-state-induction.

2 Related Work

The role of DST and DSI in task-oriented dialogue systems. Task-oriented dialogue systems are complex, traditionally involving a pipeline of multiple steps, including automatic speech recognition (ASR) [Wen *et al.*, 2017], spoken language understanding (SLU) [Qin *et al.*, 2019], dialogue state tracking (DST) [Zhong *et al.*, 2018], policy learning and natural language generation (NLG) [Chen *et al.*, 2019]. SLU consistes of two main sub-tasks, namely intent detection, which is to identify the user intent such as *hotel booking*, and slot tagging, which is to identify relevant semantic slots in a user utterance, such as `price` and `stars`. Dialogue state tracking aims to identify user goals at every turn of the dialogue, such as `inform(price=moderate, stars=4); request(phone)`, which makes the core component in a task-oriented dialogue system. Policy learning aims to learn the system action based on the current state. Natural language generation transforms the system action into natural language.

Recently, some work on task-oriented dialogue systems takes an overall end-to-end method, by encoding the user utterance and dialogue history, and then generating a response directly using a seq2seq model variant, without explicitly maintaining dialogue states [Eric and Manning, 2017; Qin *et al.*, 2020]. Compared to such work, we show that automatically inducing dialogue states can improve dialogue performance, which is consistent with observations of DST research [Lei *et al.*, 2018; Wen *et al.*, 2018]. Wen *et al.* [2017] pioneered this line of work by proposing a typical modularly connected end-to-end trainable task-oriented dialogue system directly based on text without considering the speech recognition noise and thus ignored the component of SLU to obtain the dialogue state.

DST vs SLU. In a traditional pipeline, DST operates on SLU output to update the dialogue state dealing with noise from ASR and SLU. In particular, SLU can give an N-Best list of semantic representations based on the N-Best list of sentences from ASR. DST handles all these uncertainties, e.g. error propagation, to update the dialogue state. However, the recent datasets are collected based on text without taking noisy speech inputs into consideration, which has made the task of slot tagging in SLU and the task of DST rather separately investigated. The correlation between recent slot tagging work and DST work can be subtle, and there is little discussion in the literature about their fundamental differences. In practice, widely used datasets for SLU include ATIS [Hemphill *et al.*, 1990], while for DST include MultiWoZ [Budzianowski *et al.*, 2018]. For latter datasets, there is not labeling of the SLU task. Recently, DST research has taken end-to-end methods, without considering SLU as a preprocessing step. Compared with SLU, our work is more in line with current neural DST work, with DSI being a direct alternative to DST in zero-shot training scenarios.

DSI vs robust DST. Existing work on DST integrates dialogue states by manually labeling more or less. Compared to these methods, our method has exactly the same setting as end-to-end DST, which is more cost-effective. Traditional supervised models for dialogue state tracking regard the task

Utterance 1: I would like a guesthouse rather than a star hotel.

[Chen et al, 2013; Shi et al, 2018]: `hotel_type=guesthouse, hotel_type=star hotel`

DSI (representation of user intent): `inform(hotel_type=guesthouse)`

Utterance 2: I want a flight from Chicago to Dallas.

[Chen et al, 2013; Shi et al, 2018]: `city=Chicago, city=Dallas`

DSI (more fine-grained): `inform(depature_city=Chicago, destination_city=Dallas)`

Figure 2: Comparison between DSI and previous induction methods for SLU.

as a multi-class classification problem [Mrkšić *et al.*, 2015]. Given a user utterance and existing dialogue history, a model predicts the corresponding value (or None) of each slot. However, such methods cannot predict the existence of unknown slot values. Consequently, recent work begins to investigate value generation from scratch [Ren *et al.*, 2019]. Such methods reduce the decoding complexity by avoiding the enumeration of all possible slots and values. However, the models still rely on supervised data for training. Our method employs the same decoding efficiency, yet additionally does not require labeled corpora.

DSI is also related to domain adaptations of DST in handling unknown data. Supervised methods consider multi-domain settings by parameter sharing [Wu *et al.*, 2019a]. Such methods cannot deal with unknown domains, which are dominant in practice. Some work considers zero-shot learning, transferring knowledge from known domains to unknown domains without labels [Rastogi *et al.*, 2019]. One constraint of such methods is that they rely on domain similarity for transfer, and therefore cannot be applied to general domains. In addition, they rely on schema-level slot descriptions for capturing domain correlation, which requires manual labeling and for which the quality is difficult to control. In contrast, our method can be directly used to induce dialogue states from arbitrary dialogue records.

Induction methods. The closest in spirit of our work, Chen *et al.* [2013] used the FrameNet-style frame-semantic parsers to induce slots from a user utterance; Shi *et al.* [2018] proposed a framework *auto-dialabel* to cluster noun words into slots. Compared with their work, our work is different in two main aspects. First, the problems that we solve are different, which can be seen in Figure 2. In particular, given the utterance "I would like a guesthouse rather than a star hotel.", the user intent is to book a hotel, the slots include `hotel_type=guesthouse` and `hotel_type=star hotel`, and the dialogue state is `inform(hotel_type=guesthouse)`. Given the sentence "I want a flight from Chicago to Dallas", the user intent is to book a flight, the slots include `city=Chicago` and `city=Dallas`, and the dialogue state is `inform(departure_city=Chicago, destination_city=Dallas)`. From the two examples we can see that DSI not only reflects the user goals but also is more specific to the current dialogue state, while their methods are more general to the semantics. Our task is directly useful for subsequent policy learning and response generation tasks. Second, we consider a deep neural model with hidden variables and contextualized embeddings, which also adapts better to the multi-domain scenario.

3 Task Definition: Dialogue State Induction

Given a set of customer service records without annotation (e.g. user intent and dialogue states or other manual labeling), the task is to automatically discover information that the user is looking for at each turn. We call this automatic discovery process *dialogue state induction* (DSI). In particular, at each turn, the current user utterance and dialogue history can be used as input, and the output is a set of slot-value pairs, namely, the dialogue state.

Turn-level vs joint-level dialogue state. By definition, a dialogue state should reflect the user goal from the beginning of the dialogue until the current turn. We call this a joint-level dialogue state. In practice, DST research has also investigated the extraction of local user goal at each dialogue turn, which we call a turn-level state. Our models produce dialogue state for each dialogue turn, where the input is the current user utterance and its preceding system utterance, and the output is a set of slot-value pairs. For multi-turn dialogues, the current dialogue state should reflect the whole dialogue history, as problem definition specifies. Following Zhong *et al.* [2018], we handle this issue by simply using the union of slot-value pairs in each history dialogue turn for representing the current dialogue state. When there are multiple values for one slot, we use the latest value.

4 Method

We build two incrementally more complex neural latent variable models for DSI. The models induce dialogue state over a dialogue turn according to a generate process from slots to candidate values, where candidate values are represented by both one-hot vectors and contextualized embedding vectors. The two types of representations are complementary to each other, with the latter also containing features from a global context. In particular, the current user utterance and its preceding system utterance are concatenated (in their chronical order, with the latter before the former) and fed as input to a pre-trained ELMo model to obtain the contextualized word embedding vector[1].

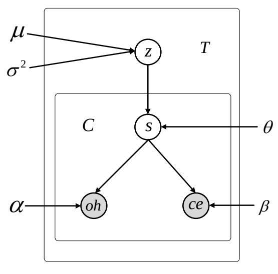
4.1 Values

Both methods are generative models that induce slot-value pairs over candidate values over a dialogue dataset. We follow Goel *et al.* [2019] and extract possible values from local conversation contexts before assigning them to slots. We take a different method to this end. In particular, we use the model of Cui and Zhang [2019] to extract POS tags and the Stanford CoreNLP toolkit to extract named entities and coreferences. A set of rules are used to extract candidate values given the POS and entity patterns including filtering stopwords, repeated candidates and non-representative entity mentions.

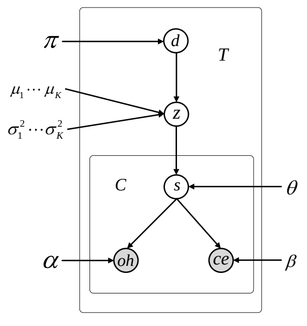
4.2 Model 1: DSI-base

Our first model is shown in Figure 3(a). The model is a generative model, which explains the occurrences of values

[1]In practice, we use the ELMo pre-trained model with the output size of 256. We also tried BERT, whose performance is almost the same as ELMo. However, BERT is much slower and more resource-intensive for training.



(a) *DSI-base*



(b) *DSI-GM*

Figure 3: Illustration of DSI models. (T – # of user turns; C – # of candidates per turn. d and $\mathbf{s}$ are discrete latent variables, $\mathbf{z}$ is a continuous latent variable. $\mathbf{oh}$ and $\mathbf{ce}$ are observed data.)

(as represented by both a one-hot vector and a contextualized dense vector) from a structured slot frame, distributed according to a hidden vector variable $\mathbf{z}$. This framework follows the variational dense embedding method of Kingma and Welling [2014]. In particular, $\mathbf{z}$ is treated as a Gaussian vector, with the mean and variance factors themselves being decided according to the observed values. As a result, its training follows a variational auto-encoder scheme.

Given a corpus of user turns $\mathcal{T}$, for each turn $t \in \mathcal{T}$, there is a set of candidate values C_t. We first sample a latent state vector $\mathbf{z}$ from a global Gaussian distribution. Then a neural network $f_s(\mathbf{z};\boldsymbol{\theta})$ takes $\mathbf{z}$ as input to encode slot distribution logits, sampling a discrete slot assignment vector $\mathbf{s}$ corresponding to each candidate $c \in C_t$ in turn t. Last, for each candidate, we sample a one-hot vector $\mathbf{oh}$ from a categorical distribution parameterized by the output of a neural network $f_{oh}(\mathbf{s};\boldsymbol{\alpha})$, as well as a contextualized word embedding vector $\mathbf{ce}$ from a multivariate Gaussian distribution parameterized by the output of a neural network $f_{ce}(\mathbf{s};\boldsymbol{\beta})$.

In particular, for the latent state $\mathbf{z}$, the Gaussian distribution is parameterized by a mean vector $\boldsymbol{\mu}$ and a diagonal covariance matrix $\boldsymbol{\sigma}^2$. Suppose that there are S slots. The categorical slot distribution for each candidate c is parameterized by a probability vector $\boldsymbol{\gamma} \in \mathbb{R}^S$. For each slot s, the Gaussian distribution for the contextualized embedding $\mathbf{ce}$ is parameterized by a mean vector $\boldsymbol{\mu}_s \in \mathbb{R}^n$ and a diagonal vector of covariance matrix $\boldsymbol{\sigma}_s \in \mathbb{R}^n$, where n represents the dimension of $\mathbf{ce}$. Further, the categorical distribution for the one-hot vector $\mathbf{oh}$ is parameterized by a probability vector $\boldsymbol{\lambda}_s \in \mathbb{R}^V$, where V is the candidate value vocabulary size. All the parameters are obtained through neural networks.

The generative process is shown in Algorithm 1. Accordingly, the joint probability for a turn t can be factorized as:

$$p(t) = p(\mathbf{z}) \prod_{c \in C_t} p(\mathbf{s}|\mathbf{z}) p(\mathbf{oh}|\mathbf{s}) p(\mathbf{ce}|\mathbf{s}) \tag{1}$$

where the probability terms are defined as:

$$p(\mathbf{z}) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}, \boldsymbol{\sigma}^2\right) \tag{2}$$
$$p(\mathbf{s}|\mathbf{z}) = \text{Cat}(\mathbf{s}|\boldsymbol{\gamma}) \tag{3}$$
$$p(\mathbf{oh}|\mathbf{s}) = \text{Cat}(\mathbf{oh}|\boldsymbol{\lambda}_s) \tag{4}$$
$$p(\mathbf{ce}|\mathbf{s}) = \mathcal{N}\left(\mathbf{ce}|\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s^2\right) \tag{5}$$

Algorithm 1 *DSI-base*

```
 1: for each user turn $t \in \mathcal{T}$ do
 2:     Sample a latent state vector $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}, \boldsymbol{\sigma}^2)$
 3:     for each candidate $c \in C_t$ do
 4:         Compute a probability vector $\boldsymbol{\gamma} = f_s(\mathbf{z}; \boldsymbol{\theta})$
 5:         Sample a slot vector $\mathbf{s} \sim$ Cat($\boldsymbol{\gamma}$)
 6:         Compute a probability vector $\boldsymbol{\lambda}_s = f_{oh}(\mathbf{s}; \boldsymbol{\alpha})$
 7:         Sample a one-hot vector $\mathbf{oh} \sim$ Cat($\boldsymbol{\lambda}_s$)
 8:         Compute $[\boldsymbol{\mu}_s; \log \boldsymbol{\sigma}_s^2] = f_{ce}(\mathbf{s}; \boldsymbol{\beta})$
 9:         Sample a contextualized word embedding vector
            $\mathbf{ce} \sim \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s^2)$
10:     end for
11: end for
```

Algorithm 2 *DSI-GM*

```
 1: for each user turn $t \in \mathcal{T}$ do
 2:     Sample a domain $d \sim$ Cat($\boldsymbol{\pi}$)
 3:     Sample a latent state vector $\mathbf{z} \sim \mathcal{N}(\boldsymbol{\mu}_d, \boldsymbol{\sigma}_d^2)$
 4:     for each candidate $c \in C_t$ do
 5:         Compute a probability vector $\boldsymbol{\gamma} = f_s(\mathbf{z}; \boldsymbol{\theta})$
 6:         Sample a slot vector $\mathbf{s} \sim$ Cat($\boldsymbol{\gamma}$)
 7:         Compute a probability vector $\boldsymbol{\lambda}_s = f_{oh}(\mathbf{s}; \boldsymbol{\alpha})$
 8:         Sample a one-hot vector $\mathbf{oh} \sim$ Cat($\boldsymbol{\lambda}_s$)
 9:         Compute $[\boldsymbol{\mu}_s; \log \boldsymbol{\sigma}_s^2] = f_{ce}(\mathbf{s}; \boldsymbol{\beta})$
10:         Sample a contextualized word embedding vector
            $\mathbf{ce} \sim \mathcal{N}(\boldsymbol{\mu}_s, \boldsymbol{\sigma}_s^2)$
11:     end for
12: end for
```

$\boldsymbol{\mu}$ and $\boldsymbol{\sigma}^2$ are the parameters of a Gaussian prior distribution. $\boldsymbol{\gamma}$, $\boldsymbol{\lambda}_s$, $\boldsymbol{\mu}_s$ and $\boldsymbol{\sigma}_s^2$ are calculated as:

$$\boldsymbol{\gamma} = \text{softmax}(W_\gamma \mathbf{z} + b_\gamma) \tag{6}$$
$$\boldsymbol{\lambda}_s = \text{softmax}(W_\lambda \mathbf{s} + b_\lambda) \tag{7}$$
$$\boldsymbol{\mu}_s = \text{BN}(W_\mu \mathbf{s} + b_\mu) \tag{8}$$
$$\boldsymbol{\sigma}_s = \text{BN}(W_\sigma \mathbf{s} + b_\sigma) \tag{9}$$

4.3 Model 2: DSI-GM

A limitation of *DSI-base* is that sampling a latent state vector $\mathbf{z}$ from a global Gaussian distribution does not sufficiently model the fact that different domains may have different distributions. For those slots that appear in different domains (for example, slot *name* appear both in domain *restaurant* and *hotel* with similar utterance contexts), it can be difficult for *DSI-base* to distinguish them correctly. We apply a Gaussian Mixture Model (GMM) in *DSI-base* by assuming the state vector is generated from a *Mixture-of-Gaussians*, in which each Gaussian represents a domain. This is inspired by Variational Deep Embedding (VaDE) [Jiang *et al.*, 2017], which combines VAE and GMM for a clustering task.

Specifically, suppose that there are K domains. As shown in Figure 3(b), we first sample a domain d from a categorical distribution parameterized by $\boldsymbol{\pi} \in \mathbb{R}^K$, where π_d is the prior probability for domain d and $\sum_{d=1}^{K} \pi_d = 1$. The latent state vector $\mathbf{z}$ is sampled from a Gaussian distribution with parameters of a mean vector $\boldsymbol{\mu}_d$ and a covariance vector $\boldsymbol{\sigma}_d$ corresponding to the chosen domain d. The latent state vector is then used for sampling a slot vector for each candidate in a turn, in the same way as *DSI-base*.

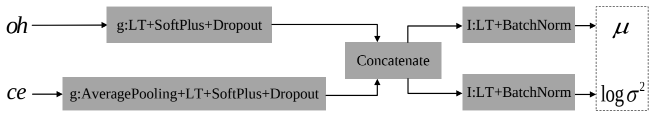


Figure 4: An encoder network is used to maximize the ELBO of *DSI-base* and *DSI-GM*. “LT” denotes linear transformation.

According to the generative process shown in Algorithm 2, the joint probability for a user turn t is

$$p(t) = p(d)p(\mathbf{z}|d) \prod_{c \in C_t} p(\mathbf{s}|\mathbf{z})p(\mathbf{oh}|\mathbf{s})p(\mathbf{ce}|\mathbf{s}), \quad (10)$$

where the additional probabilities are defined as:

$$p(d) = \mathrm{Cat}(d|\boldsymbol{\pi}) \quad (11)$$

$$p(\mathbf{z}|d) = \mathcal{N}\left(\mathbf{z}|\boldsymbol{\mu}_d, \boldsymbol{\sigma}_d^2\right) \quad (12)$$

4.4 Inference

Given the generative process of *DSI-GM*, following Liu *et al.* [2019a], we collapse the discrete slot latent variable s and rewrite the joint log-likelihood as:

$$\log p(t) = \log \int_{\mathbf{z}} \sum_{d} p(d)p(\mathbf{z}|d) \prod_{c \in C_t} p(\mathbf{oh}|\mathbf{z})p(\mathbf{ce}|\mathbf{z})d\mathbf{z}$$

$$\geq E_{q(\mathbf{z},d|t)}[\log \frac{p(t,\mathbf{z},d)}{q(\mathbf{z},d|t)}] = \mathcal{L}_{\mathrm{ELBO}}(t) \quad (13)$$

where $\mathcal{L}_{\mathrm{ELBO}}$ is the evidence lower bound (ELBO). Since direct optimization and inference for Equation 13 is intractable. We follow previous work [Kingma and Welling, 2014; Jiang *et al.*, 2017] and use a variational posterior distribution $q(\mathbf{z}, d|t)$ to approximate the true posterior distribution $p(\mathbf{z}, d|t)$. By assuming a mean field distribution [Xing *et al.*, 2003], $q(\mathbf{z}, d|t)$ can be factorized as $q(\mathbf{z}|t)q(d|t)$. The posterior $q(\mathbf{z}|t)$ can be modeled using a multivariate Gaussian distribution, with the mean vector $\boldsymbol{\mu}_d$ and the variance vector $\boldsymbol{\sigma}_d$ obtained through a neural network as shown in Figure 4. $q(d|t)$ is calculated as follows:

$$q(d|t) = p(d|\mathbf{z}) \equiv \frac{p(d)p(\mathbf{z}|d)}{\sum_{d'=1}^{K} p(d')p(\mathbf{z}|d')} \quad (14)$$

Using the reparameterization trick [Kingma and Welling, 2014], the ELBO can be decomposed into a *reconstruction* term and a *regularization* term, respectively:

$$\mathcal{L}_{\mathrm{ELBO}}(t) = E_{q(\mathbf{z},d|t)}[\log p(t|\mathbf{z})] - D_{KL}(q(\mathbf{z},d|t)||p(\mathbf{z},d)) \quad (15)$$

where $D_{KL}(q(\mathbf{z},d|t)||p(\mathbf{z},d))$ is the KL divergence between the *Mixture-of-Gaussians* prior $p(\mathbf{z},d)$ and variational posterior $q(\mathbf{z},d|t)$.

After training, the domain and slot assignment for each candidate c can be obtained by Equation 14 and as follows:

$$p(\mathbf{s}|c,t) = p(\mathbf{s}|c,\mathbf{z}) \propto p(\mathbf{s},\mathbf{oh},\mathbf{ce},\mathbf{z}) \quad (16)$$

$$= p(\mathbf{s}|\mathbf{z})p(\mathbf{oh}|\mathbf{s})p(\mathbf{ce}|\mathbf{s}) \quad (17)$$

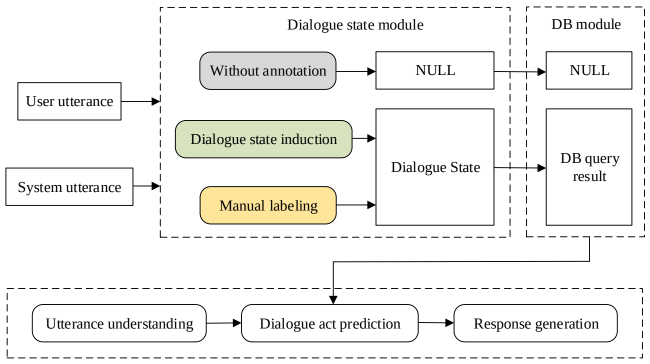


Figure 5: DSI-based dialogue response generation.

5 DSI-Based Response Generation

We apply the DSI models on the task of downstream response generation, following the pipeline system which was first proposed by Wen *et al.* [2017] and then decomposed into two components by Chen *et al.* [2019]. As shown in Figure 5, the top component consists of the dialogue state module and database operation module, while the bottom component contains the dialogue act prediction module and response generation module. For the bottom component, we directly take the recently proposed HDSA model [Chen *et al.*, 2019] to test our induced dialogue states.

For the top component, we investigate three different settings of the dialogue state module: (i) empty dialogue states under the condition that no annotation is available (grey oval), (ii) dialogue states induced by our mixture model (green oval), (iii) dialogue states obtained by manual labeling (golden oval). In the last two settings, the dialogue states are regarded as input for subsequent modules, while in the first setting, no state information is given, which corresponds to end-to-end models [Eric and Manning, 2017; Wu *et al.*, 2019b] for dialogue system. The first setting is used to test the effectiveness of our model, while the third setting can be regarded as the oracle upper bound of our model.

6 Experiments

We evaluate our proposed DSI task and its effectiveness on the downstream tasks using the MultiWOZ 2.1 [Eric *et al.*, 2019] dataset, which fixes some noisy state annotations in the MultiWOZ 2.0 [Budzianowski *et al.*, 2018] dataset. MultiWOZ2.1 contains 10,438 multi-turn dialogues and we follow the same partition as Wu *et al.* [2019a]. To justify the generalization of the proposed model, we also use a recently proposed SGD [Rastogi *et al.*, 2019] dataset, which contains 16,142 multi-turn dialogues and is the largest existing conversational corpus. We use the same train/validation split sets as Rastogi *et al.* [2019].

6.1 Experimental Settings

For the DSI models, the Adam optimizer is used to maximize the ELBO of Equation 15. All the models are trained over the training set, where hyper-parameters are tuned on the development set, before being finally used on the test set. Since no manual labels are available, we follow Liu *et al.* [2019a] and

Models	MultiWOZ 2.1								SGD							
	Turn level				Joint level				Turn level				Joint level			
	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy
Random	1.49	1.51	1.49	1.39	0.21	0.28	0.23	0.02	0.94	0.95	0.94	0.92	0.05	0.08	0.06	0.02
DSI-base	38.8	37.7	37.3	25.7	33.9	32.1	32.1	2.3	27.0	26.0	26.0	21.1	13.9	17.5	14.5	2.3
DSI-GM	52.5	39.3	49.6	36.1	49.2	43.2	44.8	5.0	34.7	33.4	33.5	27.5	19.0	22.9	19.5	3.1

Table 1: Overall results of DSI.

Name	Value	Name	Value
Domain number (*DSI-GM* only)	100	Batch size	200
Slot number (*DSI-base*/*DSI-GM*)	300/1000	Dropout	0.2
Feature dimension	256	Learning rate	0.02
Linear transformation layer size	100	Momentum	0.99

Table 2: Hyper-parameters settings.

Dialogue State	Dialog Act Prediction			Delexicalized	
	Precision	Recall	F1	BLEU	Entity F1
None	71.0	67.4	69.1	18.7	54.6
DSI-GM	72.0	70.5	71.2	20.8	56.5
Manual labeling	75.6	73.0	74.2	21.6	61.3

Table 3: Empirical results on MultiWOZ dialogue act prediction and response generation.

select the hyper-parameters which fit the best ELBO score on the dev set as shown in Table 2. For the downstream HDSA model, we directly take the original hyper-parameters. Since our datasets are manually labeled with domains and slots (*restaurant-name*), we can name the induced slots after the gold-standard slots that have the maximum value match. In addition, in the datasets, each slot is labeled with a service domain, and thus we obtain a domain output also.

6.2 Evaluation Metrics

DSI For each turn, *DSI-base* and *DSI-GM* induce several or no slot-value pairs based on the current user utterance and its preceding system utterance. We compare the DSI outputs with the slots that have non-empty assignments in the ground truth dialogue states for the current user turn. We consider the following two metrics[2]:

1. **State Matching** (*Precision*, *Recall* and *F1-score* in Table 1): Similar to previous work [Liu *et al.*, 2019a], we use state matching to evaluate the overlapping of induced states and the ground truth.
2. **Goal Accuracy** (*Accuracy* in Table 1): We adopt this standard metric from DST [Wu *et al.*, 2019a; Zhong *et al.*, 2018]. The predicted dialogue states for a turn is considered as true only when all the user search goal constraints are correctly and exactly identified.

We evaluate both metrics in both the **turn level** and the **joint level** (Table 1). The joint level metrics are more strict in jointly considering the output of all turns.

Response generation Following Chen *et al.* [2019], the dialogue act prediction results are evaluated in terms of *Precision*, *Recall* and *F1-score*. Delexicalized-BLEU and Entity F1 are used to evaluate response generation.

6.3 Results

DSI Performance The DSI results are shown in Table 1. We have four main observations:

- Both *DSI* models show great advantages over a random select strategy, which randomly assigns a reference slot for each candidate. This shows the strength of our neural generative models with hidden variables.

[2]A fuzzy matching mechanism is used to compare induced values with the ground truth.

- *DSI-GM* outperforms *DSI-base* on both the turn level and joint level metrics, which demonstrates the effectiveness of the GMM model, which finds an appropriate domain first before sampling a latent state representation. We attribute this to the fact that the dialogue states can be more effectively regarded as a hierarchical structure (i.e., domain-slot-value) and hence first detecting a domain and then a slot under this domain can help alleviate the difficulty of distinguishing the appropriate slot in a large mixture of similar slot values.
- The *joint goal accuracy* is significantly lower compared with the other metrics, which shows that the metric can be overly strict in our *unsupervised* setting. This is a consistent observation of recent work on cross-lingual dialogue state tracking [Liu *et al.*, 2019b], which shows that the joint goal accuracy of a cross-lingual DST model can be as low as 11% on accuracy even with cross-lingual contextualized embeddings. Furthermore, the *joint goal F1-score* can reach 44.8% on MultiWOZ dataset, which shows that our model can achieve promising performance without any labeled training data.
- The results among all metrics on the SGD dataset are lower than those on the MultiWOZ2.1 dataset. We attribute it to this reason that the SGD dataset is more difficult than the MultiWOZ dataset because it contains more types of domains and slots (16 domains and 214 slots as compared with 7 domains and 24 slots in MultiWOZ).

DAP and Response Generation The dialogue act prediction (DAP) and response generation results are shown in Table 3. Compared to not using dialogue states, the outputs of *DSI-GM* allows a subsequent model [Chen *et al.*, 2019] to improve the F1-score of dialogue act prediction by 2.1%, and further improve the BLEU and entity F1 for a system output utterance by 2.1% and 1.9%, respectively. This shows that maintaining dialogue states can be useful in neural dialogue systems, as consistent with observations from DST research [Lei *et al.*, 2018; Wen *et al.*, 2018]. The results demonstrate that DSI is a useful task in dialogue systems research and our baseline models are effective. In Table 3, the gap between *DSI-GM* and *manual labeling* indicates further rooms for improvement on the dialogue model.

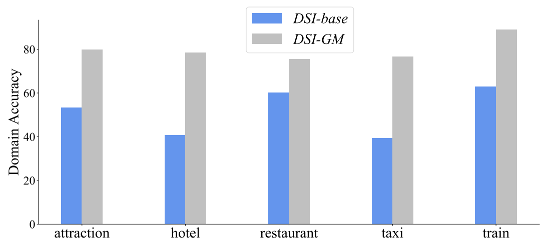
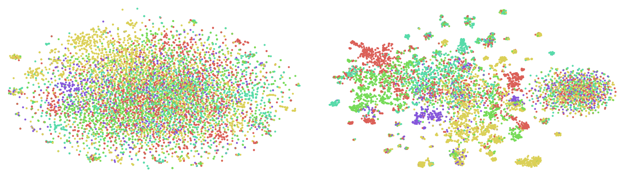
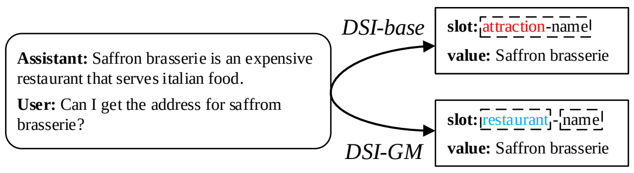


Figure 6: Domain accuracy.



(a) *DSI-base* (b) *DSI-GM*

Figure 7: Visualization of state vectors learned by *DSI-base* and *DSI-GM*. Each color represents a domain.



Assistant: Saffron brasserie is an expensive restaurant that serves italian food.

User: Can I get the address for saffrom brasserie?

Figure 8: Example output by *DSI-base* and *DSI-GM*. The cyan domain is correct, while the red domain is wrong.

6.4 *DSI-GM* vs. *DSI-base*

Domain Accuracy We measure the fraction of user turns for which the domains are correctly induced for all candidates. Figure 6 shows the comparison between the *DSI-base* model and the *DSI-GM* model. We can see that the *DSI-GM* outperforms the *DSI-base* on each domain, which demonstrates that explicitly modeling the domain distribution is effective across all domains. In addition, to better intuitively know how much better the domain representation can be modeled through *DSI-GM*, we visualize the learned state representations of *DSI-base* and *DSI-GM* only with its domain label. In particular, we use t-SNE to reduce the dimensionality of the latent representation **z** and plot the whole test set in Figure 7. Compared with *DSI-base*, whose latent state representations are mixed in the domain level, *DSI-GM* shows its superiority in representing different domains, as representations are clustering in a more orderly way. This further demonstrates the effectiveness of *GMM* applied in our task.

Case Study Figure 8 shows a case on the dialogue states induced by *DSI-base* and *DSI-GM*. In this case, both the *attraction* and *restaurant* domains consist of a *name* slot, which shares similar contexts such as "Can i get the address for". This can make their contextualized features similar although the two *name* slots should be treated as two distinct types. *DSI-base* generates an incorrect domain *attraction* while *DSI-GM* induces the correct domain *restaurant*. We attribute it to the fact that the *DSI-GM* model captures domain information by explicitly modeling the multi-domain distribution through a *Mixture-of-Gaussians* instead of a global *Gaussian*.

	attraction	hotel	restaurant	taxi	train
DSI-base	27.9	21.7	26.1	30.7	26.0
DSI-GM	40.3	31.4	35.6	39.9	36.8

Table 4: Turn goal accuracy per domain.

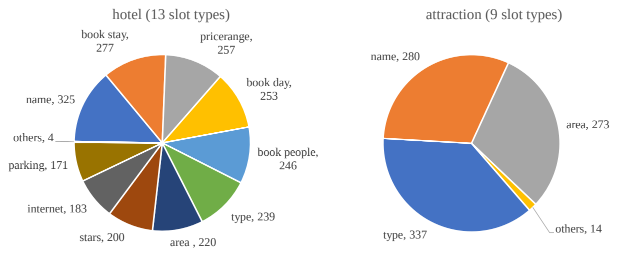


Figure 9: Slots distributions on *hotel* and *attraction* domains.

6.5 Error Analysis

We present the accuracies on each domain between *DSI-base* and *DSI-GM*. The results are shown in Table 4. Both *DSI-base* and *DSI-GM* give the lowest accuracy on the *hotel* domain and a relatively higher accuracy on the *attraction* domain. To further understand the reason, we calculate the statistics of slots on the *hotel* and *attraction* domains, which are shown in Figure 9. It can be seen that the numbers of dominant slot types of the two domains are 10 and 3, respectively, correctly recognizing which can give strong overall accuracies. In both domains, these slots are distributed evenly. This indicates that the number of distinct slots is a key factor to the difficulty level for our DSI models, which is intuitive.

7 Conclusion

We proposed a novel task of *dialogue state induction*, which is to automatically identify dialogue state slots and values over a large set of dialogue records. Compared with existing research, our task is practically more useful for handling the large variety of services available in the industry, which disallows scalable manual labeling of dialogue states. We further built two neural generative models with latent variables. Results on standard DST datasets show that the models can effectively induce meaningful dialogue states from raw dialogue data, and further improve the results of a dialogue system compared to without using dialogue states. Our methods can serve as baselines for further research on the task.

Acknowledgments

We thank the anonymous reviewers for their detailed and constructive comments. The first three authors contributed equally. Yue Zhang is the corresponding author. We would like to acknowledge funding support from National Natural Science Foundation of China under Grant No.61976180 and the Westlake University and Bright Dream Joint Institute for Intelligent Robotics.

References

[Budzianowski *et al.*, 2018] Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gašić. MultiWOZ - a large-scale multi-domain wizard-of-Oz dataset for task-oriented dialogue modelling. In *EMNLP*, 2018.

[Chen *et al.*, 2013] Yun-Nung Chen, William Yang Wang, and Alexander I Rudnicky. Unsupervised induction and filling of semantic slots for spoken dialogue systems using frame-semantic parsing. In *ASRU Workshop*, 2013.

[Chen *et al.*, 2019] Wenhu Chen, Jianshu Chen, Pengda Qin, Xifeng Yan, and William Yang Wang. Semantically conditioned dialog response generation via hierarchical disentangled self-attention. In *ACL*, 2019.

[Cui and Zhang, 2019] Leyang Cui and Yue Zhang. Hierarchically-refined label attention network for sequence labeling. In *EMNLP*, 2019.

[Eric and Manning, 2017] Mihail Eric and Christopher D Manning. A copy-augmented sequence-to-sequence architecture gives good performance on task-oriented dialogue. In *EACL*, 2017.

[Eric *et al.*, 2019] Mihail Eric, Rahul Goel, Shachi Paul, Abhishek Sethi, Sanchit Agarwal, Shuyag Gao, and Dilek Hakkani-Tur. Multiwoz 2.1: Multi-domain dialogue state corrections and state tracking baselines. *arXiv*, 2019.

[Goel *et al.*, 2019] Rahul Goel, Shachi Paul, and Dilek Hakkani-Tür. Hyst: A hybrid approach for flexible and accurate dialogue state tracking. In *Interspeech*, 2019.

[Hemphill *et al.*, 1990] Charles T Hemphill, John J Godfrey, and George R Doddington. The atis spoken language systems pilot corpus. In *HLT*, 1990.

[Jiang *et al.*, 2017] Zhuxi Jiang, Yin Zheng, Huachun Tan, Bangsheng Tang, and Hanning Zhou. Variational deep embedding: An unsupervised and generative approach to clustering. In *IJCAI*, 2017.

[Kingma and Welling, 2014] Diederik P. Kingma and Max Welling. Auto-encoding variational bayes. In *ICLR*, 2014.

[Lei *et al.*, 2018] Wenqiang Lei, Xisen Jin, Min-Yen Kan, Zhaochun Ren, Xiangnan He, and Dawei Yin. Sequicity: Simplifying task-oriented dialogue systems with single sequence-to-sequence architectures. In *ACL*, 2018.

[Liu *et al.*, 2019a] Xiao Liu, Heyan Huang, and Yue Zhang. Open domain event extraction using neural latent variable models. In *ACL*, 2019.

[Liu *et al.*, 2019b] Zihan Liu, Genta Indra Winata, Zhaojiang Lin, Peng Xu, and Pascale Fung. Attention-informed mixed-language training for zero-shot cross-lingual task-oriented dialogue systems. In *AAAI*, 2019.

[Mrkšić *et al.*, 2015] Nikola Mrkšić, Diarmuid Ó Séaghdha, Blaise Thomson, Milica Gasic, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. Multi-domain dialog state tracking using recurrent neural networks. In *ACL*, 2015.

[Qin *et al.*, 2019] Libo Qin, Wanxiang Che, Yangming Li, Haoyang Wen, and Ting Liu. A stack-propagation framework with token-level intent detection for spoken language understanding. In *EMNLP*, 2019.

[Qin *et al.*, 2020] Libo Qin, Xiao Xu, Wanxiang Che, Yue Zhang, and Ting Liu. Dynamic Fusion Network for Multi-Domain End-to-end Task-Oriented Dialog. *arXiv*, 2020.

[Rastogi *et al.*, 2019] Abhinav Rastogi, Xiaoxue Zang, Srinivas Sunkara, Raghav Gupta, and Pranav Khaitan. Towards scalable multi-domain conversational agents: The schema-guided dialogue dataset. In *AAAI*, 2019.

[Ren *et al.*, 2019] Liliang Ren, Jianmo Ni, and Julian McAuley. Scalable and accurate dialogue state tracking via hierarchical sequence generation. In *EMNLP*, 2019.

[Shi *et al.*, 2018] Chen Shi, Qi Chen, Lei Sha, Sujian Li, Xu Sun, Houfeng Wang, and Lintao Zhang. Auto-dialabel: Labeling dialogue data with unsupervised learning. In *EMNLP*, 2018.

[Wen *et al.*, 2017] Tsung-Hsien Wen, David Vandyke, Nikola Mrkšić, Milica Gasic, Lina M Rojas Barahona, Pei-Hao Su, Stefan Ultes, and Steve Young. A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, 2017.

[Wen *et al.*, 2018] Haoyang Wen, Yijia Liu, Wanxiang Che, Libo Qin, and Ting Liu. Sequence-to-sequence learning for task-oriented dialogue with dialogue state representation. In *COLING*, 2018.

[Williams *et al.*, 2014] Jason D Williams, Matthew Henderson, Antoine Raux, Blaise Thomson, Alan Black, and Deepak Ramachandran. The dialog state tracking challenge series. *AI Magazine*, 2014.

[Wu *et al.*, 2019a] Chien-Sheng Wu, Andrea Madotto, Ehsan Hosseini-Asl, Caiming Xiong, Richard Socher, and Pascale Fung. Transferable multi-domain state generator for task-oriented dialogue systems. In *ACL*, 2019.

[Wu *et al.*, 2019b] Chien-Sheng Wu, Richard Socher, and Caiming Xiong. Global-to-local memory pointer networks for task-oriented dialogue. In *ICLR*, 2019.

[Xing *et al.*, 2003] Eric P Xing, Michael I Jordan, and Stuart Russell. A generalized mean field algorithm for variational inference in exponential families. In *UAI*, 2003.

[Young *et al.*, 2010] Steve Young, Milica Gašić, Simon Keizer, François Mairesse, Jost Schatzmann, Blaise Thomson, and Kai Yu. The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *CSL*, 2010.

[Zhang *et al.*, 2019] Jian-Guo Zhang, Kazuma Hashimoto, Chien-Sheng Wu, Yao Wan, Philip S Yu, Richard Socher, and Caiming Xiong. Find or classify? dual strategy for slot-value predictions on multi-domain dialog state tracking. *arXiv*, 2019.

[Zhong *et al.*, 2018] Victor Zhong, Caiming Xiong, and Richard Socher. Global-locally self-attentive encoder for dialogue state tracking. In *ACL*, 2018.