Adversarial Domain Adaptation for Variational Neural Language Generation in Dialogue Systems

Van-Khanh Tran[1,2] and Le-Minh Nguyen[1]

[1]Japan Advanced Institute of Science and Technology, JAIST
1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
{tvkhanh, nguyenml}@jaist.ac.jp
[2]University of Information and Communication Technology, ICTU
Thai Nguyen University, Vietnam
tvkhanh@ictu.edu.vn

Abstract

Domain Adaptation arises when we aim at learning from source domain a model that can perform acceptably well on a different target domain. It is especially crucial for Natural Language Generation (NLG) in Spoken Dialogue Systems when there are sufficient annotated data in the source domain, but there is a limited labeled data in the target domain. How to effectively utilize as much of existing abilities from source domains is a crucial issue in domain adaptation. In this paper, we propose an adversarial training procedure to train a Variational encoder-decoder based language generator via multiple adaptation steps. In this procedure, a model is first trained on a source domain data and then fine-tuned on a small set of target domain utterances under the guidance of two proposed critics. Experimental results show that the proposed method can effectively leverage the existing knowledge in the source domain to adapt to another related domain by using only a small amount of in-domain data.

1 Introduction

Traditionally, Spoken Dialogue Systems are typically developed for various specific domains, including: finding a hotel, searching a restaurant (Wen et al., 2015a), or buying a tv, laptop (Wen et al., 2015b), flight reservations (Levin et al., 2000), etc. Such system are often requiring a well-defined ontology, which is essentially a data structured representation that the dialogue system can converse about. Statistical approaches to multi-domain in SDS system have shown promising results in how to reuse data in a domain-scalable framework efficiently (Young et al., 2013). Mrkšić et al. (2015) addressed the question of multi-domain in the SDS belief tracking by training a general model and adapting it to each domain.

Recently, Recurrent Neural Networks (RNNs) based methods have shown improving results in tackling the domain adaptation issue (Chen et al., 2015; Shi et al., 2015; Wen et al., 2016a; Wen et al., 2016b). Such generators have also achieved promising results when providing such adequate annotated datasets (Wen et al., 2015b; Wen et al., 2015a; Tran et al., 2017; Tran and Nguyen, 2017a; Tran and Nguyen, 2017b). More recently, the development of the variational autoencoder (VAE) framework (Kingma and Welling, 2013; Rezende and Mohamed, 2015) has paved the way for learning large-scale, directed latent variable models. This has brought considerable benefits to significant progress in natural language processing (Bowman et al., 2015; Miao et al., 2016; Purushotham et al., 2017; Mnih and Gregor, 2014), dialogue system (Wen et al., 2017; Serban et al., 2017).

This paper presents an adversarial training procedure to train a variational neural language generator via multiple adaptation steps, which enables the generator to learn more efficiently when in-domain data is in short supply. In summary, we make the following contributions: (1) We propose a variational approach for an NLG problem which benefits the generator to adapt faster to new, unseen domain irrespective of scarce target resources; (2) We propose two critics in an adversarial training procedure, which can guide the generator to generate outputs that resemble the sentences drawn from the target domain; (3) We propose a unifying variational domain adaptation architecture which performs acceptably well

in a new, unseen domain by using a limited amount of in-domain data; (4) We investigate the effectiveness of the proposed method in different scenarios, including ablation, domain adaptation, scratch, and unsupervised training with various amount of data.

2 Related Work

Generally, Domain Adaptation involves two different types of datasets, one from a source domain and the other from a target domain. The source domain typically contains a sufficient amount of annotated data such that a model can be efficiently built, while there is often little or no labeled data in the target domain. Domain adaptation for NLG have been less studied despite its important role in developing multi-domain SDS. Walker et al. (2001) proposed a SPoT-based generator to address domain adaptation problems. Subsequently, a system focused on tailoring user preferences (Walker et al., 2007), and controlling user perceptions of linguistic style (Mairesse and Walker, 2011). Moreover, a phrase-based statistical generator (Mairesse et al., 2010) using graphical models and active learning, and a multi-domain procedure (Wen et al., 2016a) via data counterfeiting and discriminative training.

Neural variational framework for generative models of text have been studied longitudinally. Chung et al. (2015) proposed a recurrent latent variable model VRNN for sequential data by integrating latent random variables into hidden state of a RNN model. A hierarchical multi scale recurrent neural networks was proposed to learn both hierarchical and temporal representation (Chung et al., 2016). Zhang et al. (2016) introduced a variational neural machine translation that incorporated a continuous latent variable to model underlying semantics of sentence pairs. Bowman et al. (2015) presented a variational autoencoder for unsupervised generative language model.

Adversarial adaptation methods have shown promising improvement in many machine learning applications despite the presence of domain shift or dataset bias, which reduce the difference between the training and test domain distributions, and thus improve generalization performance. Tzeng et al. (2017) proposed an improved unsupervised domain adaptation method to learn a discriminative mapping of target images to the source feature space by fooling a domain discriminator that tries to differentiate the encoded target images from source examples. We borrowed the idea of (Ganin et al., 2016), where a domain-adversarial neural network are proposed to learn features that are discriminative for the main learning task on the source domain, and indiscriminate with respect to the shift between domains.

3 Variational Domain-Adaptation Neural Language Generator

Drawing inspiration from Variational autoencoder (Kingma and Welling, 2013) with assumption that there exists a continuous latent variable z from a underlying semantic space of Dialogue Act (DA) and utterance pairs $(\mathbf{d}, \mathbf{y})$, we explicitly model the space together with variable $\mathbf{d}$ to guide the generation process, $i.e.$, $p(\mathbf{y}|z, \mathbf{d})$. With this assumption, the original conditional probability evolves to reformulate as follows:

$$p(\mathbf{y}|\mathbf{d}) = \int_z p(\mathbf{y}, z|\mathbf{d})\mathbf{d}_z = \int_z p(\mathbf{y}|z, \mathbf{d})p(z|\mathbf{d})\mathbf{d}_z \tag{1}$$

This latent variable enables us to model the underlying semantic space as a global signal for generation, in which the variational lower bound of variational generator can be formulated as follows:

$$\mathcal{L}_{VAE}(\theta, \phi, \mathbf{d}, \mathbf{y}) = -KL(q_\phi(z|\mathbf{d}, \mathbf{y})||p_\theta(z|\mathbf{d})) + \mathbb{E}_{q_\phi(z|\mathbf{d}, \mathbf{y})}[\log p_\theta(\mathbf{y}|z, \mathbf{d})] \tag{2}$$

where: $p_\theta(z|\mathbf{d})$ is the prior model, $q_\phi(z|\mathbf{d}, \mathbf{y})$ is the posterior approximator, and $p_\theta(\mathbf{y}|z, \mathbf{d})$ is the decoder with the guidance from global signal z, $KL(Q||P)$ is the Kullback-Leibler divergence between Q and P.

3.1 Variational Neural Encoder

The variational neural encoder aims at encoding a given input sequence $w_1, w_2, .., w_L$ into continuous vectors. In this work, we use a 1-layer, Bidirectional LSTM (BiLSTM) to encode the sequence embedding. The BiLSTM consists of forward and backward LSTMs, which read the sequence from left-to-right and right-to-left to produce both forward and backward sequence of hidden states $(\overrightarrow{\mathbf{h}_1}, .., \overrightarrow{\mathbf{h}_L})$, and
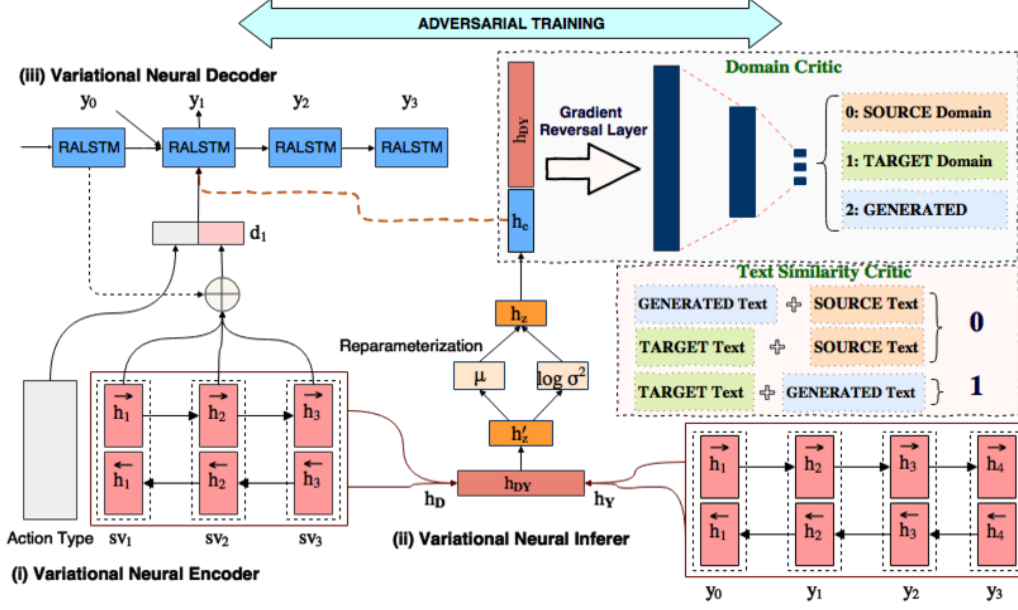
Figure 1: The VDANLG architecture which consists of two main components: the VRALSTM to generate the sentence and two Critics with an adversarial training procedure to guide the model in domain adaptation.

$(\overleftarrow{\mathbf{h}}_1, .., \overleftarrow{\mathbf{h}}_L)$, respectively. We then obtain the sequence of encoded hidden states $\mathbf{h_E} = (\mathbf{h}_1, \mathbf{h}_2, .., \mathbf{h}_L)$ where: $\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i + \overleftarrow{\mathbf{h}}_i$. We utilize this encoder to represent both the sequence of slot-value pairs $\{\mathbf{sv}_i\}_{i=1}^{T_{DA}}$ in a given Dialogue Act, and the corresponding utterance $\{\mathbf{y}_i\}_{i=1}^{T_Y}$ (see the red parts in Figure 1). We finally operate the mean-pooling over the BiLSTM hidden vectors to obtain the representation: $\mathbf{h}_D = \frac{1}{T_{DA}} \sum_i^{T_{DA}} \mathbf{h}_i, \mathbf{h}_Y = \frac{1}{T_Y} \sum_i^{T_Y} \mathbf{h}'_i$. The encoder, accordingly, produces both the DA representation vector which flows into the inferer and decoder, and the utterance representation which streams to the posterior approximator.

3.2 Variational Neural Inferer

In this section, we describe our approach to model both the prior $p_\theta(z|\mathbf{d})$ and the posterior $q_\phi(z|\mathbf{d}, \mathbf{y})$ by utilizing neural networks.

Neural Posterior Approximator

Modeling the true posterior $p(z|\mathbf{d}, \mathbf{y})$ is usually intractable. Traditional approach fails to capture the true posterior distribution of z due to its oversimplified assumption when using the mean-field approaches. Following the work of (Kingma and Welling, 2013), in this paper we employ neural network to approximate the posterior distribution of z to simplify the posterior inference. We assume the approximation has the following form:

$$q_\phi(z|\mathbf{d}, \mathbf{y}) = \mathcal{N}(z; \mu(f(\mathbf{h}_D, \mathbf{h}_Y)), \sigma^2(f(\mathbf{h}_D, \mathbf{h}_Y))\boldsymbol{I}) \tag{3}$$

where: the mean μ and standard variance σ are the outputs of the neural network based on the representations of $\mathbf{h}_D$ and $\mathbf{h}_Y$. The function f is a non-linear transformation that project the both DA and utterance representations onto the latent space:

$$\mathbf{h}'_z = f(\mathbf{h}_D, \mathbf{h}_Y) = g(\mathbf{W}_z[\mathbf{h}_D; \mathbf{h}_Y] + b_z) \tag{4}$$

where: $\mathbf{W}_z \in \mathbb{R}^{d_z \times (d_{\mathbf{h}_D} + d_{\mathbf{h}_Y})}$, $b_z \in \mathbb{R}^{d_z}$ are matrix and bias parameters respectively, d_z is the dimensionality of the latent space, $g(.)$ is an elements-wise activation function which we set to be $Relu$ in our experiments. In this latent space, we obtain the diagonal Gaussian distribution parameter μ and $\log \sigma^2$ through linear regression:

$$\mu = \mathbf{W}_\mu \mathbf{h}'_z + b_\mu, \log \sigma^2 = \mathbf{W}_\sigma \mathbf{h}'_z + b_\sigma \tag{5}$$

where: $\mu, \log \sigma^2$ are both d_z dimension vectors.

Neural Prior Model

We model the prior as follows:

$$p_\theta(z|\mathbf{d}) = \mathcal{N}(z; \mu'(\mathbf{d}), \sigma'(\mathbf{d})^2 \boldsymbol{I}) \tag{6}$$

where: μ' and σ' of the prior are neural models based on DA representation only, which are the same as those of the posterior $q_\phi(z|\mathbf{d}, \mathbf{y})$ in Eq. 4 and Eq. 5, except for the absence of $\mathbf{h}_Y$. To acquire a representation of the latent variable z, we utilize the same technique as proposed in VAE (Kingma and Welling, 2013) and re-parameterize it as follows:

$$\mathbf{h}_z = \mu + \sigma \odot \epsilon, \epsilon \sim \mathcal{N}(0, \boldsymbol{I}) \tag{7}$$

In addition, we set $\mathbf{h}_z$ to be the mean of the prior $p_\theta(z|\mathbf{d})$, *i.e.*, μ', during decoding due to the absence of the utterance y. Intuitively, by parameterizing the hidden distribution this way, we can back-propagate the gradient to the parameters of the encoder and train the whole network with stochastic gradient descent. Note that the parameters for the prior and the posterior are independent of each other.

In order to integrate the latent variable $\mathbf{h}_z$ into the decoder, we use a non-linear transformation to project it onto the output space for generation:

$$\mathbf{h}_e = g(\mathbf{W}_e \mathbf{h}_z + b_e) \tag{8}$$

where: $\mathbf{h}_e \in \mathbb{R}^{d_e}$. It is important to notice that due to the sample noise ϵ, the representation of $\mathbf{h}_e$ is not fixed for the same input DA and model parameters. This benefits the model to learn to quickly adapt to a new domain (see Table 1-(a) and Table 3, *sec.* 3).

3.3 Variational Neural Decoder

Given a DA $\mathbf{d}$ and the latent variable z, the decoder calculates the probability over the generation $\mathbf{y}$ as a joint probability of ordered conditionals:

$$p(\mathbf{y}|z, \mathbf{d}) = \prod_{j=1}^{T_Y} p(\mathbf{y}_t|\mathbf{y}_{<t}, z, \mathbf{d}) \tag{9}$$

where: $p(\mathbf{y}_t|\mathbf{y}_{<t}, z, \mathbf{d}) = g'(RNN(\mathbf{y}_t, \mathbf{h}_{t-1}, \mathbf{d}_t)$ In this paper, we borrow the $\mathbf{d}_t$ calculation and the computational RNN cell from (Tran and Nguyen, 2017a) where RNN(.)=RALSTM(.) with a slightly modification in order to integrate the representation of latent variable, *i.e.*, $\mathbf{h}_e$, into the RALSTM cell, which is denoted by the bold dashed orange arrow in Figure 1-(iii). We modify the cell calculation as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W}_{4d_h, 4d_h} \begin{pmatrix} \mathbf{h}_e \\ \mathbf{d}_t \\ \mathbf{h}_{t-1} \\ \mathbf{y}_t \end{pmatrix} \tag{10}$$

where: $\mathbf{i}_i, \mathbf{f}_t, \mathbf{o}_t$ are input, forget and output gates respectively, d_h is hidden layer size, $\mathbf{W}_{4d_h, 4d_h}$ is model parameter.

The resulting Variational RALSTM (VRALSTM) model is demonstrated in Figure 1-(i), (ii), (iii), in which the latent variable can affect the hidden representation through the gates. This allows the model can indirectly take advantage of the underlying semantic information from the latent variable z. In addition, when the model learns to adapt to a new domain with unseen dialogue act, the semantic representation $\mathbf{h}_e$ can help to guide the generation process (see *sec.* 6.3 for details).

3.4 Critics

In this section, we introduce a *text-similarity critic* and a *domain critic* to guarantee, as much as possible, that the generated sentences resemble the sentences drawn from the target domain.

Text similarity critic

To check the relevance between sentence pair in two domains and to encourage the model generating sentences in the style which is highly *similar* to those in the target domain, we propose a Text Similarity Critic (SC) to classify $(\mathbf{y}_{(1)}, \mathbf{y}_{(2)})$ as 1-similar or 0-unsimilar text style. The model SC consists of two parts: a shared BiLSTM $\mathbf{h}_Y$ with the Variational Neural Encoder to represent the $\mathbf{y}_{(1)}$ sentence, and a second BiLSTM to encode the $\mathbf{y}_{(2)}$ sentence. The SC model takes input as a pair $(\mathbf{y}_{(1)}, \mathbf{y}_{(2)})$ of ([*target*], *source*), ([*target*], *generated*), and ([*generated*], *source*). Note that we give priority to encoding the $\mathbf{y}_{(1)}$ sentence in [.] using the shared BiLSTM, which guides the model to learn the sentence style from the target domain, and also contributes the target domain information into the global latent variables. We further utilize Siamese recurrent architectures (Neculoiu et al., 2016) for learning sentence similarity, in which the architecture allows us to learn useful representations with limited supervision.

Domain critic

In consideration of the shift between domains, we introduce a Domain Critic (DC) to classify sentence as *source*, *target*, or *generated* domain, respectively. Drawing inspiration from work of (Ganin et al., 2016), we model DC with a gradient reversal layer and two standard feed-forward layers. It is important to notice that our DC model shares parameters with the Variational Neural Encoder and the Variational Neural Inferer. The DC model takes input as a pair of given DA and corresponding utterance to produce a concatenation of both its representation and its latent variable in the output space, which is then passed through a feed-forward layer and a 3-labels classifier. In addition, the gradient reversal layer, which multiplies the gradient by a specific negative value during back-propagation training, ensures that the feature distributions over the two domains are made similar, as indistinguishable as possible for the domain critic, hence resulting in the domain-invariant features.

4 Training Domain Adaptation Model

Given a training instance represented by a pair of DA and sentence $(\mathbf{d}^{(i)}, \mathbf{y}^{(i)})$ from the rich source domain $\mathcal{S}$ and the limited target domain $\mathcal{T}$, the task aims at finding a set of parameters $\Theta_{\mathcal{T}}$ that can perform acceptably well on the target domain.

4.1 Training Critics

We provide as following the training objective of SC and DC. For SC, the goal is to classify a sentence pair into 1-*similar* or 0-*unsimilar* textual style. This procedure can be formulated as a supervised classification training objective function:

$$\mathcal{L}_s(\psi) = -\sum_{n=1}^{N} \log C_s(l_s^n | \mathbf{y}_{(1)}^n, \mathbf{y}_{(2)}^n, \psi), l_s^n = \begin{cases} 1 - similar & \text{if } (\mathbf{y}_{(1)}^n, \mathbf{y}_{(2)}^n) \in \mathcal{P}_{sim}, \\ 0 - unsimilar & \text{if } (\mathbf{y}_{(1)}^n, \mathbf{y}_{(2)}^n) \in \mathcal{P}_{unsim}, \end{cases} \quad (11)$$

$$\mathcal{Y}_{\mathcal{G}} = \{\mathbf{y} | \mathbf{y} \sim \mathcal{G}(. | \mathbf{d}_{\mathcal{T}}, .)\}, \mathcal{P}_{sim} = \{\mathbf{y}_{\mathcal{T}}^n, \mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n\}, \mathcal{P}_{unsim} = (\{\mathbf{y}_{\mathcal{T}}^n, \mathbf{y}_{\mathcal{S}}^n\}, \{\mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n, \mathbf{y}_{\mathcal{S}}^n\})$$

where: N is number of sentences, ψ is the model parameters of SC, $\mathcal{Y}_{\mathcal{G}}$ denotes sentences generated from the current generator $\mathcal{G}$ given target domain dialogue act $\mathbf{d}_{\mathcal{T}}$. The scalar probability $C_s(1 | \mathbf{y}_{\mathcal{T}}^n, \mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n)$ indicates how a generated sentence $\mathbf{y}_{\mathcal{Y}_{\mathcal{G}}}^n$ is relevant to a target sentence $\mathbf{y}_{\mathcal{T}}^n$.

The DC critic aims at classifying a pair of DA-utterance into *source*, *target*, or *generated* domain. This can also be formulated as a supervised classification training objective as follows:

$$\mathcal{L}_d(\varphi) = -\sum_{n=1}^{N} \log C_d(l_d^n | \mathbf{d}^n, \mathbf{y}^n, \varphi), l_d^n = \begin{cases} source & \text{if } (\mathbf{d}^n, \mathbf{y}^n) \in (\mathcal{D}_{\mathcal{S}}, \mathcal{Y}_{\mathcal{S}}), \\ target & \text{if } (\mathbf{d}^n, \mathbf{y}^n) \in (\mathcal{D}_{\mathcal{T}}, \mathcal{Y}_{\mathcal{T}}), \\ generated & \text{if } (\mathbf{d}^n, \mathbf{y}^n) \in (\mathcal{D}_{\mathcal{T}}, \mathcal{Y}_{\mathcal{G}}), \end{cases} \quad (12)$$

where: φ is the model parameters of DC, $(\mathcal{D}_S, \mathcal{Y}_S)$, $(\mathcal{D}_T, \mathcal{Y}_T)$ are the DA-utterance pairs from source, target domain, respectively. Note also that the scalar probability $C_d(target | \mathbf{d}^n, \mathbf{y}^n)$ indicates how likely the DA-utterance pair $(\mathbf{d}^n, \mathbf{y}^n)$ is from the target domain.

4.2 Training Variational Generator

We utilize the Monte Carlo method to approximate the expectation over the posterior in Eq. 2, *i.e.*, $\mathbb{E}_{q_\phi(z|\mathbf{d},\mathbf{y})}[.] \simeq \frac{1}{M}\sum_{m=1}^{M}\log p_\theta(\mathbf{y}|\mathbf{d},\mathbf{h}_z^{(m)})$ where: M is the number of samples. In this study, the joint training objective for a training instance $(\mathbf{d},\mathbf{y})$ is formulated as follows:

$$\mathcal{L}(\theta,\phi) \simeq -KL(q_\phi(z|\mathbf{d},\mathbf{y})||p_\theta(z|\mathbf{d})) + \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T_y}\log p_\theta(\mathbf{y}_t|\mathbf{y}_{<t},\mathbf{d},\mathbf{h}_z^{(m)}) \tag{13}$$

where: $\mathbf{h}_z^{(m)} = \mu + \sigma \odot \epsilon^{(m)}$, and $\epsilon^{(m)} \sim \mathcal{N}(0,\mathbf{I})$. The first term is the KL divergence between two Gaussian distribution, and the second term is the approximation expectation. We simply set $M = 1$ which degenerates the second term to the objective of conventional generator. Since the objective function in Eq. 13 is differentiable, we can jointly optimize the parameter θ and variational parameter ϕ using standard gradient ascent techniques.

4.3 Adversarial Training

Our domain adaptation architecture is demonstrated in Figure 1, in which both generator $\mathcal{G}$ and critics C_s, and C_d jointly train by pursuing competing goals as follows. Given a dialogue act $\mathbf{d}_\mathcal{T}$ in the target domain, the generator generates K sentences $\mathbf{y}$'s. It would prefer a "*good*" generated sentence $\mathbf{y}$ if the values of $C_d(target|\mathbf{d}_\mathcal{T},\mathbf{y})$ and $C_s(1|\mathbf{y}_\mathcal{T},\mathbf{y})$ are large. In contrast, the critics would prefer large values of $C_d(generated|\mathbf{d}_\mathcal{T},\mathbf{y})$ and $C_s(1|\mathbf{y},\mathbf{y}_\mathcal{S})$, which imply the small values of $C_d(target|\mathbf{d}_\mathcal{T},\mathbf{y})$ and $C_s(1|\mathbf{y}_\mathcal{T},\mathbf{y})$. We propose a domain-adversarial training procedure in order to iteratively updating the generator and critics as described in Algorithm 1. While the parameters of generator are optimized to minimize their loss in the training set, the parameters of the critics are optimized to minimize the error of text similarity, and to maximize the loss of domain classifier.

Algorithm 1: Adversarial Training Procedure

Require: generator $\mathcal{G}$, domain critic C_d, text similarity critic C_s, generated sentence $\mathcal{Y}_\mathcal{G} = \emptyset$;
Input: DA-utterance pairs of source $(\mathcal{D}_\mathcal{S},\mathcal{Y}_\mathcal{S})$, target $(\mathcal{D}_\mathcal{T},\mathcal{Y}_\mathcal{T})$;
1 Pretrain $\mathcal{G}$ on $(\mathcal{D}_\mathcal{S},\mathcal{Y}_\mathcal{S})$ using VRALSTM;
2 **while** Θ *has not converged* **do**
3   **for** *i = 0, .., $N_\mathcal{T}$* **do**
4   Sample $(\mathbf{d}_\mathcal{S},\mathbf{y}_\mathcal{S})$ from source domain;
5   (D_1)-Compute $g_d = \nabla_\varphi \mathcal{L}_d(\varphi)$ using Eq. 12 for $(\mathbf{d}_\mathcal{S},\mathbf{y}_\mathcal{S})$ and $(\mathbf{d}_\mathcal{T},\mathbf{y}_\mathcal{T})$;
6   (D_2)-Adam update of φ for C_d using g_d;
7   (G_1)-Compute $g_\mathcal{G} = \{\nabla_\theta \mathcal{L}(\theta,\phi), \nabla_\phi \mathcal{L}(\theta,\phi)\}$ using Eq. 13
8   (G_2)-Adam update of θ,ϕ for $\mathcal{G}$ using $g_\mathcal{G}$
9   (S_1)-Compute $g_s = \nabla_\psi \mathcal{L}_s(\psi)$ using Eq. 11 for $(\mathbf{y}_\mathcal{T},\mathbf{y}_\mathcal{S})$;
10   (S_2)-Adam update of ψ for C_s using g_s;
11        $\mathcal{Y}_\mathcal{G} \leftarrow \{\mathbf{y}_{\bar{k}}\}_{k=1}^{K}$, where $\mathbf{y}_{\bar{k}} \sim \mathcal{G}(.|\mathbf{d}_\mathcal{T}^{(i)},.)$;
12   Choose top k best sentences of $\mathcal{Y}_\mathcal{G}$;
13   **for** *j = 1,..,k* **do**
14   (D_1), (D_2) steps for C_d with $(\mathbf{d}_\mathcal{T},\mathbf{y}_\mathcal{G}^{(j)})$;
15   (S_1), (S_2) steps for C_s with $(\mathbf{y}_\mathcal{G}^{(j)},\mathbf{y}_\mathcal{S})$ and $(\mathbf{y}_\mathcal{T},\mathbf{y}_\mathcal{G}^{(j)})$;
16   **end**
17   **end**
18 **end**

Generally, the current generator $\mathcal{G}$ for each training iteration i takes a *target* dialogue act $\mathbf{d}_\mathcal{T}^{(i)}$ as input to over-generate a set $\mathcal{Y}_\mathcal{G}$ of K candidate sentences (step 11). We then choose top **k** best sentences in the $\mathcal{Y}_\mathcal{G}$ set (step 12) after re-ranking to measure how "*good*" the generated sentences are by using the critics (steps 14-15). These "*good*" signals from the critics can guide the generator step by step to generate the outputs which resemble the sentences drawn from the target domain. Note that the re-ranking step is important for separating the "*correct*" sentences from the current generated outputs $\mathcal{Y}_\mathcal{G}$ by penalizing the generated sentences which have redundant or missing slots.

5 Experiments

We conducted experiments on the proposed models in different scenarios: ***Adaptation***, ***Scratch***, and ***All*** using several model architectures, evaluation metrics, datasets (Wen et al., 2016a), and configurations (see Appendix A).

KL cost annealing strategy (Bowman et al., 2015) encourages the model to encode meaningful representations into the latent vector z, in which we gradually anneal the KL term from 0 to 1. This helps our model to achieve solutions with non-zero KL term.
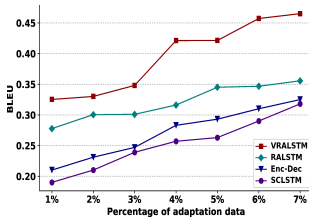
Gradient reversal layer (Ganin et al., 2016) leaves the input unchanged during forward propagation and reverses the gradient by multiplying it with a negative scalar $-\lambda_p$ during the backpropagation-based training. We set the domain adaptation parameter λ_p which gradually increases, starting from 0 to 1, by using the following schedule for each training step i: $p = float(i)/num\_steps$, and $\lambda_p = \frac{2}{1+exp(-10*p)} - 1$ where: num_steps is a constant which is set to be 8600, p is training progress. This strategy allows the Domain critic to be less sensitive to noisy signal at the early training stages.

6 Results and Analysis

6.1 Integrating Variational Inference

We compare the original model RALSTM with its modification by integrating Variational Inference (VRALSTM) as demonstrated in Table 2 and Table 1-(a). It clearly shows that the VRALSTM not only preserves the power of the original RALSTM on generation task since its performances are very competitive to those of RALSTM, but also provides a compelling evidence on adapting to a new, unseen domain when the target domain data is scarce, *i.e.*, from 1% to 7%. Table 3, *sec.* 3 further shows the necessity of the integrating in which the VRALSTM achieved a significant improvement over the RALSTM in ***Scratch*** scenario, and of the adversarial domain adaptation algorithm in which although both the RALSTM and VRALSTM model can perform well when providing sufficient in-domain training data (Table 2), the performances are extremely impaired when training from ***Scratch*** with only a limited data. These indicate that the proposed variational method can learn the underlying semantic of DA-utterance pairs in the source domain via the representation of the latent variable z, from which when adapting to another domain, the models can leverage the existing knowledge to guide the generation process.

Table 1: Results when adapting models trained on (a) union, and (b) counterfeting dataset.

Source \ Target(*Test*)	R2H(*Hotel*)		H2R(*Restaurant*)		L2T(*Tv*)		T2L(*Laptop*)	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
Hotel♯	-	-	0.5931	12.50%	0.4183	2.38%	0.3426	13.02%
Restaurant♯	0.6224	1.99%	-	-	0.4211	2.74%	0.3540	13.13%
Tv♯	0.6153	4.30%	0.5835	14.49%	-	-	0.3630	7.44%
Laptop♯	0.6042	5.22%	0.5598	15.61%	0.4268	1.05%	-	-

(a) Result on **Laptop** when adapting models trained on [Restaurant+Hotel] data.

(b) Results evaluated on (*Test*) domains by ***Unsupervised*** adapting VDANLG from Source domains using only **10%** of the **Target** domain **Counterfeit X2Y**. {**X,Y**}=**R** : Restaurant, **H** : Hotel, **T** : Tv, **L** : Laptop.

Table 2: Results evaluated on **Target** domains by training models from scratch with ***All*** in-domain data.

Model \ Target	Hotel		Restaurant		Tv		Laptop	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
HLSTM (Wen et al., 2015a)	0.8488	2.79%	0.7436	0.85%	0.5240	2.65%	0.5130	1.15%
SCLSTM (Wen et al., 2015b)	0.8469	3.12%	0.7543	0.57%	0.5235	2.41%	0.5109	0.89%
Enc-Dec (Wen et al., 2016b)	0.8537	4.78%	0.7358	2.98%	0.5142	3.38%	0.5101	4.24%
RALSTM (Tran and Nguyen, 2017a)	0.8911	0.48%	0.7739	0.19%	0.5376	0.65%	0.5222	0.49%
VRALSTM (Ours)	0.8851	0.57%	0.7709	0.36%	0.5356	0.73%	0.5210	0.59%

Table 3: Ablation studies' results evaluated on **Target** domains by *adaptation* training proposed models from Source domains using only *10%* amount of the **Target** domain data (*sec.* 1, 2, 4, 5). The results were averaged over 5 randomly initialized networks.

	Target / Source	Hotel BLEU	Hotel ERR	Restaurant BLEU	Restaurant ERR	Tv BLEU	Tv ERR	Laptop BLEU	Laptop ERR
no Critics	Hotel	-	-	0.6814	11.62%	0.4968	12.19%	0.4915	3.26%
	Restaurant	0.7983	8.59%	-	-	0.4805	13.70%	0.4829	9.58%
	Tv	0.7925	12.76%	0.6840	8.16%	-	-	0.4997	4.79%
	Laptop	0.7870	15.17%	0.6859	7.55%	0.4953	18.60%	-	-
	[R+H]	-	-	-	-	0.5019	7.43%	0.4977	5.96%
	[L+T]	0.7935	11.71%	0.6927	6.49%	-	-	-	-
+ DC + SC	Hotel	-	-	0.7131	2.53%	0.5164	3.25%	0.5007	1.68%
	Restaurant	0.8217	3.95%	-	-	0.5043	2.99%	0.4931	2.77%
	Tv	0.8251	4.89%	0.6971	4.62%	-	-	0.5009	2.10%
	Laptop	0.8218	2.89%	0.6926	2.87%	0.5243	1.52%	-	-
	[R+H]	-	-	-	-	0.5197	2.58%	0.5009	1.61%
	[L+T]	0.8252	2.87%	0.7066	3.73%	-	-	-	-
scr10	RALSTM	0.6855	22.53%	0.6003	17.65%	0.4009	22.37%	0.4475	24.47%
	VRALSTM	0.7378	15.43%	0.6417	15.69%	0.4392	17.45%	0.4851	10.06%
+ DC only	Hotel	-	-	0.6823	4.97%	0.4322	27.65%	0.4389	26.31%
	Restaurant	0.8031	6.71%	-	-	0.4169	34.74%	0.4245	26.71%
	Tv	0.7494	14.62%	0.6430	14.89%	-	-	0.5001	15.40%
	Laptop	0.7418	19.38%	0.6763	9.15%	0.5114	10.07%	-	-
	[R+H]	-	-	-	-	0.4257	31.02%	0.4331	31.26%
	[L+T]	0.7658	8.96%	0.6831	11.45%	-	-	-	-
+ SC only	Hotel	-	-	0.6976	5.00%	0.4896	9.50%	0.4919	9.20%
	Restaurant	0.7960	4.24%	-	-	0.4874	12.26%	0.4958	5.61%
	Tv	0.7779	10.75%	0.7134	5.59%	-	-	0.4913	13.07%
	Laptop	0.7882	8.08%	0.6903	11.56%	0.4963	7.71%	-	-
	[R+H]	-	-	-	-	0.4950	8.96%	0.5002	5.56%
	[L+T]	0.7588	9.53%	0.6940	10.52%	-	-	-	-

sec. 3: Training RALSTM and VRALSTM models from *scratch* using 10% of **Target** domain data;

6.2 Ablation Studies

The ablation studies (Table 3, *sec.* 1, 2) demonstrate the contribution of two Critics, in which the models were assessed with either no Critics or both or only one. It clearly sees that combining both Critics makes a substantial contribution to increasing the BLEU score and decreasing the slot error rate by a large margin in every dataset pairs. A comparison of model adapting from source Laptop domain between VRALSTM without Critics (Laptop$^\flat$) and VDANLG (Laptop$^\sharp$) evaluated on the target **Hotel** domain shows that the VDANLG not only has better performance with much higher the BLEU score, 82.18 in comparison to 78.70, but also significantly reduce the ERR, from 15.17% down to 2.89%. The trend is consistent across all the other domain pairs. These stipulate the necessary Critics in effective learning to adapt to a new domain.

Table 3, *sec.* 4 further demonstrates that using DC only (*sec.* 4) brings a benefit of effectively utilizing similar slot-value pairs seen in the training data to *closer* domain pairs such as: Hotel→Restaurant (68.23 BLEU, 4.97 ERR), Restaurant→Hotel (80.31 BLEU, 6.71 ERR), Laptop→Tv (51.14 BLEU, 10.07 ERR), and Tv→Laptop (50.01 BLEU, 15.40 ERR) pairs. Whereas it is inefficient for the *longer* domain pairs since their performances are worse than those without Critics, or in some cases even worse than the VRALSTM in *scr10* scenario, such as Restaurant→Tv (41.69 BLEU, 34.74 ERR), and the cases where Laptop to be a Target domain. On the other hand, using only SC (*sec.* 5) helps the models achieve better results since it is aware of the sentence style when adapting to the target domain.

6.3 Distance of Dataset Pairs

To better understand the effectiveness of the methods, we analyze the learning behavior of the proposed model between different dataset pairs. The datasets' order of difficulty was, from easiest to hardest:

Hotel↔Restaurant↔Tv↔Laptop. On the one hand, it might be said that the *longer* datasets' distance is, the more difficult of domain adaptation task becomes. This clearly shows in Table 3, *sec.* 1, at **Hotel** column where the adaptation ability gets worse regarding decreasing the BLEU score and increasing the ERR score alongside the order of Restaurant→Tv→Laptop datasets. On the other hand, the *closer* the dataset pair is, the faster model can adapt. It can be expected that the model can better adapt to the target **Tv/Laptop** domain from source Laptop/Tv than those from source Restaurant, Hotel, and vice versa, the model can easier adapt to the target **Restaurant/Hotel** domain from source Hotel/Restaurant than those from Laptop, Tv. However, the above-mentioned is not always true that the proposed method can perform acceptably well from *easy* source domains (Hotel, Restaurant) to the more *difficult* target domains (Tv, Laptop) and vice versa (Table 3, *sec.* 1, 2).

Table 3, *sec.* 2 further shows that the proposed method is able to leverage the out of domain knowledge since the adaptation models trained on union source dataset, such as [R+H] or [L+T], show better performances than those trained on individual source domain data. A specific example in Table 3, *sec.* 2 shows that the adaptation VDANLG model trained on the source union dataset of Laptop and Tv ([L+T]$^\sharp$) has better performance, at 82.52 BLEU and 2.87 ERR, than those models trained on the individual source dataset, such as Laptop$^\sharp$ (82.18 BLEU, 2.89 ERR), and Tv$^\sharp$ (82.51 BLEU, 4.89 ERR). Another example in Table 3, *sec.* 2 also shows that the adaptation VDANLG model trained on the source union dataset of Restaurant and Hotel ([R+H]$^\sharp$) has better results, at 51.97 BLEU and 2.58 ERR, than those models trained on the separate source dataset, such as Restaurant$^\sharp$ (50.43 BLEU, 2.99 ERR), and Hotel$^\sharp$ (51.64 BLEU, 3.25 ERR). The trend is mostly consistent across all other domain comparisons in different training scenarios. All these demonstrate that the proposed model can learn global semantics that can be efficiently transferred into new domains.

6.4 *Adaptation* vs. *All* Training Scenario

It is interesting to compare ***Adaptation*** (Table 3, *sec.* 2) with ***All*** training scenario (Table 2). The VDANLG model shows its considerable ability to shift to another domain with a limited of in-domain labels whose results are competitive to or in some cases better than the previous models trained on full labels of the **Target** domain. A specific comparison evaluated on the **Tv** domain where the VDANLG model trained on the source Laptop$^\sharp$ achieved better performance, at 52.43 BLEU and 1.52 ERR, than HLSTM (52.40, 2.65), SCLSTM (52.35, 2.41), and Enc-Dec (51.42, 3.38). The VDANLG models, in many cases, also have lower of slot error rate ERR scores than the Enc-Dec model. These indicate the stable strength of the VDANLG models in adapting to a new domain when the target domain data is scarce.

6.5 Unsupervised Domain Adaptation

We further examine the effectiveness of the proposed methods by training the VDANLG models on target ***Counterfeit*** datasets (Wen et al., 2016a). The promising results are shown in Table 1-(b), despite the fact that the models were instead adaptation trained on the **Counterfeit** datasets, or in other words, were indirectly trained on the (*Test*) domains. However, the proposed models still showed positive signs in remarkably reducing the slot error rate ERR in the cases of *Hotel* and *Tv* be the (*Test*) domains. Surprisingly, even the source domains (Hotel$^\sharp$/Restaurant$^\sharp$) are far from the (*Test*) domain *Tv*, and the **Target** domain **Counterfeit L2T** is also very different to the source domains, the model can still acceptably adapt well since its BLEU scores on (*Test*) Tv domain reached to (41.83/42.11), and it also produced a very low of ERR scores (2.38/2.74). This phenomenon will be further investigated in the unsupervised scenario in the future work.

6.6 Comparison on Generated Outputs

On the one hand, the VRALSTM models (trained from ***Scratch*** or trained adapting model from Source domains) produce the outputs with a diverse range of error types, including missing, misplaced, redundant, wrong slots, or even spelling mistake information, leading to a very high of the slot error rate ERR score. Specifically, the VRALSTM from ***Scratch*** tends to make repeated slots and also many of the missing slots in the generated outputs since the training data may inadequate for the model to generally

Table 4: Comparison of top **Laptop** responses generated for different scenarios by adaptation training VRALSTM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VRALSTM from **Scratch**. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG♯ = VRALSTM♭+SC+DC.

Model	Generated Responses from Laptop Domain
DA 1	compare(name='*tecra erebus 20*'; memory='*4 gb*'; isforbusinesscomputing='*true*'; name='*satellite heracles 45*'; memory='*2 gb*'; isforbusinesscomputing='*false*')
Reference 1	compared to *tecra erebus 20* which has a *4 gb* memory and *is for business computing* , *satellite heracles 45* has a *2 gb* memory and *is not for business computing* . which one do you prefer
VRALSTM	which would be the *tecra erebus 20 is a business computing* laptop with *4 gb* of memory and is the SLOT_NAME , and is not for business computing . [satellite heracles 45][2 gb]
Hotel♭	*the tecra erebus 20 is used for business computing* . *the satellite heracles 45* has *4 gb* of memory and a SLOT_BATTERY battery life for business computing . which one do you want
Restaurant♭	*the tecra erebus 20 is for business computing* . *the satellite heracles 45* which has *4 gb* of memory and *is not for business computing* . which one do you want [2 gb]
Tv♭	*the tecra erebus 20* has *4 gb* of memory and is not for business computing . which one do you prefer [is for business computing][satellite heracles 45][2 gb]
[R+H]♭	the *tecra erebus 20* is not for business computing . which one do you want a business computing . which one do you prefer [4 gb][is for business computing][satellite heracles 45][2 gb]
Hotel♯	the *tecra erebus 20* has a *4 gb* memory , that *is for business computing* . the *satellite heracles 45* with *2 gb* of memory and *is not for business computing* . which one do you want [OK]
Restaurant♯	the *tecra erebus 20* has a *4 gb* memory , and *is for business computing* . the *satellite heracles 45 is not for business computing* . which one do you want to know more [2 gb]
Tv♯	the *tecra erebus 20 is a business computing* . the *satellite heracles 45* has a *4 gb* memory and *is not for business computing* . which one do you prefer [2 gb]
[R+H]♯	the *tecra erebus 20 is for business computing* , has a *2 gb* of memory. the *satellite heracles 45* has *4 gb* of memory , *is not for business computing*. which one do you want

handle the unseen dialog acts. Whereas the VRALSTM models without Critics adapting trained from Source domains (denoted by ♭ in Table 4 and Appendix B. Table 5) tend to generate the outputs with fewer error types than the model from **Scratch** because the VRALSTM♭ models may capture the overlap slots of both source and target domain during adaptation training.

On the other hand, under the guidance of the Critics (SC and DC) in an adversarial training procedure, the VDANLG model (denoted by ♯) can effectively leverage the existing knowledge of the source domains to better adapt to the target domains. The VDANLG models can generate the outputs in style of the target domain with much fewer the error types compared with the two above models. Moreover, the VDANLG models seem to produce satisfactory utterances with more correct generated slots. For example, a sample outputted by the [R+H]♯ in Table 4-example 1 contains all the required slots with only a misplaced information of two slots *2 gb* and *4 gb*, while the generated output produced by Hotel♯ is a successful generation. Another samples in Appendix B. Table 5 generated by the Hotel♯, Tv♯, [R+H]♯ (in DA 2) and Laptop♯ (DA 3) models are all fulfilled responses. An analysis of the generated responses in Table 5-example 2 illustrates that the VDANLG models seem to generate a concise response since the models show a tendency to form some potential slots into a concise phrase, *i.e.*, "SLOT_NAME SLOT_TYPE". For example, the VDANLG models tend to concisely response as "the *portege phosphorus 43 laptop* ..." instead of "the *portege phosphorus 43* is a *laptop* ...". All these above demonstrate that the VDANLG models have ability to produce better results with a much lower of the slot error rate ERR score.

7 Conclusion and Future Work

We have presented an integrating of a variational generator and two Critics in an adversarial training algorithm to examine the model ability in domain adaptation task. Experiments show that the proposed models can perform acceptably well in a new, unseen domain by using a limited amount of in-domain data. The ablation studies also demonstrate that the variational generator contributes to effectively learn the underlying semantic of DA-utterance pairs, while the Critics show its important role of guiding the model to adapt to a new domain. The proposed models further show a positive sign in unsupervised domain adaptation, which would be a worthwhile study in the future.

1214

Acknowledgements

References

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*.

Xie Chen, Tian Tan, Xunying Liu, Pierre Lanchantin, Moquan Wan, Mark JF Gales, and Philip C Woodland. 2015. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. In *Sixteenth Annual Conference of the International Speech Communication Association*.

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.

Junyoung Chung, Sungjin Ahn, and Yoshua Bengio. 2016. Hierarchical multiscale recurrent neural networks. *arXiv preprint arXiv:1609.01704*.

Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, and Victor Lempitsky. 2016. Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

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

Esther Levin, Shrikanth Narayanan, Roberto Pieraccini, Konstantin Biatov, Enrico Bocchieri, Giuseppe Di Fabbrizio, Wieland Eckert, Sungbok Lee, A Pokrovsky, Mazin Rahim, et al. 2000. The at&t-darpa communicator mixed-initiative spoken dialog system. In *Sixth International Conference on Spoken Language Processing*.

François Mairesse and Marilyn A. Walker. 2011. Controlling user perceptions of linguistic style: Trainable generation of personality traits. *Comput. Linguist.*, 37(3):455–488, September.

François Mairesse, Milica Gašić, Filip Jurčíček, Simon Keizer, Blaise Thomson, Kai Yu, and Steve Young. 2010. Phrase-based statistical language generation using graphical models and active learning. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, ACL '10, pages 1552–1561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Yishu Miao, Lei Yu, and Phil Blunsom. 2016. Neural variational inference for text processing. In *International Conference on Machine Learning*, pages 1727–1736.

Andriy Mnih and Karol Gregor. 2014. Neural variational inference and learning in belief networks. *arXiv preprint arXiv:1402.0030*.

Nikola Mrkšić, Diarmuid O Séaghdha, Blaise Thomson, Milica Gašić, Pei-Hao Su, David Vandyke, Tsung-Hsien Wen, and Steve Young. 2015. Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*.

Paul Neculoiu, Maarten Versteegh, Mihai Rotaru, and Textkernel BV Amsterdam. 2016. Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318. Association for Computational Linguistics.

Sanjay Purushotham, Wilka Carvalho, Tanachat Nilanon, and Yan Liu. 2017. Variational adversarial deep domain adaptation for health care time series analysis.

Danilo Jimenez Rezende and Shakir Mohamed. 2015. Variational inference with normalizing flows. *arXiv preprint arXiv:1505.05770*.

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.

Yangyang Shi, Martha Larson, and Catholijn M Jonker. 2015. Recurrent neural network language model adaptation with curriculum learning. *Computer Speech & Language*, 33(1):136–154.

Van-Khanh Tran and Le-Minh Nguyen. 2017a. Natural language generation for spoken dialogue system using rnn encoder-decoder networks. In *Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL 2017)*, pages 442–451, Vancouver, Canada, August. Association for Computational Linguistics.

Van-Khanh Tran and Le-Minh Nguyen. 2017b. Semantic refinement gru-based neural language generation for spoken dialogue systems. *arXiv preprint arXiv:1706.00134*.

Van-Khanh Tran, Le-Minh Nguyen, and Satoshi Tojo. 2017. Neural-based natural language generation in dialogue using rnn encoder-decoder with semantic aggregation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 231–240, Saarbrcken, Germany, August. Association for Computational Linguistics.

Eric Tzeng, Judy Hoffman, Kate Saenko, and Trevor Darrell. 2017. Adversarial discriminative domain adaptation. *arXiv preprint arXiv:1702.05464*.

Marilyn A Walker, Owen Rambow, and Monica Rogati. 2001. Spot: A trainable sentence planner. In *Proceedings of the 2nd NAACL*, pages 1–8. Association for Computational Linguistics.

Marilyn A Walker, Amanda Stent, François Mairesse, and Rashmi Prasad. 2007. Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Tsung-Hsien Wen, Milica Gašić, Dongho Kim, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015a. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In *Proceedings SIGDIAL*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gašić, Nikola Mrkšić, Pei-Hao Su, David Vandyke, and Steve Young. 2015b. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016a. Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Tsung-Hsien Wen, Milica Gašic, Nikola Mrkšic, Lina M Rojas-Barahona, Pei-Hao Su, David Vandyke, and Steve Young. 2016b. Toward multi-domain language generation using recurrent neural networks.

Tsung-Hsien Wen, Yishu Miao, Phil Blunsom, and Steve Young. 2017. Latent intention dialogue models. *arXiv preprint arXiv:1705.10229*.

Steve Young, Milica Gašić, Blaise Thomson, and Jason D Williams. 2013. Pomdp-based statistical spoken dialog systems: A review. *Proceedings of the IEEE*, 101(5):1160–1179.

B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang. 2016. Variational Neural Machine Translation. *ArXiv e-prints*, May.

Appendix A. Experimental Setups

We followed the configurations for the RALSTM model from work of (Tran and Nguyen, 2017a), in which the hidden layer size and beam width were set to be 80 and 10, respectively, and the generators were trained with a 70% of keep dropout rate. We performed 5 runs with different random initialization of the network, and the training process is terminated by using early stopping. We then selected a model that yields the highest BLEU score (Papineni et al., 2002) on the validation set. For the variational inference, we set the latent variable size to be 16. We used Adam optimizer with the learning rate is initially set to be 0.001, and after 3 epochs for the Union dataset and 5 epochs for the single dataset the learning rate is decayed every epoch using an exponential rate of 0.95. The decoding phase is mostly similar to those in work of (Tran and Nguyen, 2017a).

Appendix B. Generated Utterances

Table 5 presents top responses generated for different scenarios from Laptop and TV domains.

Table 5: Comparison of top responses generated for different scenarios by adaptation training VRAL-STM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VRAL-STM from **Scratch**. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG♯ = VRALSTM♭+SC+DC.

Model	Generated Responses from Laptop Domain
DA 2	inform(name=*'portege phosphorus 43'*; type=*'laptop'*; design=*'black magnesium chassis with brushed metallic finish , matt black keyboard'*; isforbusinesscomputing=*'false'*; drive=*'320 gb'*)
Reference 2	the *portege phosphorus 43 laptop* has a *320 gb* drive , *is not for business computing* and *has a black magnesium chassis with brushed metallic finish , matt black keyboard*
VRALSTM	the *portege phosphorus 43* is a *laptop* with a *320 gb* drive and *has a black magnesium chassis with brushed metallic finish , matt black keyboard* . [is not for business computing]
Hotel♭	the *portege phosphorus 43* is a *laptop* has a *320 gb* drive , *is not for business computing* . it is not for business computing , it has a design of *black magnesium chassis with brushed metallic finish , matt black keyboard*
Restaurant♭	the *portege phosphorus 43* is a *laptop* with a *320 gb* drive , has a design of *black magnesium chassis with brushed metallic finish , matt black keyboard* . [is not for business computing]
Tv♭	the *portege phosphorus 43* is a *laptop* with a *black magnesium chassis with brushed metallic finish , matt black keyboard* . it is not for business computing [320 gb]
[R+H]♭	the *portege phosphorus 43* is a *laptop* with a *black magnesium chassis with brushed metallic finish , matt black keyboard* [is not used for business computing] [320 gb]
Hotel♯	the *portege phosphorus 43 laptop* has a *320 gb* drive , has a *black magnesium chassis with brushed metallic finish , matt black keyboard* design and *is not for business computing* [OK]
Restaurant♯	*the portege phosphorus 43 laptop* has a *320 gb* drive , it is for business computing , it has a design of *black magnesium chassis with brushed metallic finish , matt black keyboard*
Tv♯	*the portege phosphorus 43 laptop* has a *320 gb* drive and a design of *black magnesium chassis with brushed metallic finish , matt black keyboard* . it *is not for business computing* [OK]
[R+H]♯	the *portege phosphorus 43 laptop* has a *320 gb* drive , and *is not for business computing* . it has a *black magnesium chassis with brushed metallic finish , matt black keyboard* [OK]

Model	Generated Responses from TV Domain
DA 3	compare(name=*'crios 69'*; ecorating=*'a++'*; powerconsumption=*'44 watt'*; name=*'dinlas 61'*; ecorating=*'a+'*; powerconsumption=*'62 watt'*)
Reference 3	compared to *crios 69* which is in the *a++* eco rating and has *44 watt* power consumption , *dinlas 61* is in the *a+* eco rating and has *62 watt* power consumption . which one do you prefer ?
VRALSTM	the *crios 69* is the dinlas 61 is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME is the SLOT_NAME . it has an a++ eco rating [44 watt][a+][62 watt]
Hotel♭	the *crios 69* has a *44 watt* power consumption , whereas the *dinlas 61* has *62 watt* power consumption , whereas the SLOT_NAME has SLOT_POWERCONSUMPTION power consumption and has an a++ eco rating [a+]
Restaurant♭	the *crios 69* has a *a++* eco rating , *44 watt* power consumption , and an a+ eco rating and 62 watt power consumption [dinlas 61]
Laptop♭	the *crios 69* has SLOT_HDMIPORT hdmi port -s , the *dinlas 61* has a++ eco rating and 44 watt power consumption [62 watt][a+]
[R+H]♭	the *crios 69* is in the SLOT_FAMILY product family with a++ eco rating ? [44 watt][dinlas 61][62 watt][a+]
Hotel♯	the *crios 69* has an *a++* eco rating and *44 watt* power consumption and a 62 watt power consumption [dinlas 61][a+]
Restaurant♯	the *crios 69* has *44 watt* power consumption of *a++* and has an a+ eco rating and 62 watt power consumption [dinlas 61]
Laptop♯	the *crios 69* has an *a++* eco rating and *44 watt* power consumption , whereas the *dinlas 61* has *62 watt* power consumption and *a+* eco rating . [OK]
[R+H]♯	the *crios 69* has *44 watt* power consumption , and an *a++* eco rating and the *dinlas 61* has a *62 watt* power consumption . [a+]