Computer Speech & Language 65 (2021) 101120

Contents lists available at ScienceDirect

Computer Speech & Language

journal homepage: www.elsevier.com/locate/csl

Variational model for low-resource natural language generation in spoken dialogue systems☆

Van-Khanh Tran[a,b], Le-Minh Nguyen*,[a]

[a] Japan Advanced Institute of Science and Technology, JAIST 1-1 Asahidai, Nomi, Ishikawa, 923-1292, Japan
[b] University of Information and Communication Technology, ICTU Thai Nguyen University, Vietnam

ARTICLE INFO

Article History:
Received 10 August 2018
Revised 15 May 2020
Accepted 28 May 2020
Available online 7 June 2020

Keywords:
Neural language generation
Domain adaptation
Low-resource data
Variational autoencoder
Deconvolutional neural network
CNN
RNN
LSTM

ABSTRACT

Natural Language Generation (NLG) plays a critical role in Spoken Dialogue Systems (SDSs), aims at converting a meaning representation into natural language utterances. Recent deep learning-based generators have shown improving results irrespective of providing sufficient annotated data. Nevertheless, how to build a generator that can effectively utilize as much of knowledge from a low-resource setting data is a crucial issue for NLG in SDSs. This paper presents a variational-based NLG framework to tackle the NLG problem of having limited annotated data in two scenarios, domain adaptation and low-resource in-domain training data. Based on this framework, we propose a novel adversarial domain adaptation NLG taclking the former issue, while the latter issue is also handled by a second proposed dual variational model. We extensively conducted the experiments on four different domains in a variety of training scenarios, in which the experimental results show that the proposed methods not only outperform previous methods when having sufficient training dataset but also show its ability to work acceptably well when there is a small amount of in-domain data or adapt quickly to a new domain with only a low-resource target 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). Such these system are often requiring a well-defined ontology, which is essentially a data structured representation that the dialogue system can converse about. The processes for collecting such well-defined ontology datasets are extremely expensive and time-consuming. Furthermore, the models have obtained very good performance irrespective of providing adequate labeled datasets in the supervised learning manner, while *low-resource* setting data easily results in worse performance models. There are two potential solutions for this problem, which are domain adaptation training and model designing for low-resource training manner.

First, *domain adaptation* training which aims at learning from sufficient source domain a model that can perform acceptably well on a different target domain with a limited labeled target data. Domain adaptation generally 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 the target domain is assumed to have different characteristics

☆ Part of this paper has been published as two conference papers (Tran and Nguyen, 2018a; 2018b) in COLING 2018 and CoNLL 2018. This version extends previous works by incorporating some modifications, which brings in a unified generation framework, in-depth analysis, justification of the approach, and extensive experiments with the variational models on both scenarios of having low-resource setting data.

*Corresponding author.
E-mail addresses: tvkhanh@jaist.ac.jp, tvkhanh@ictu.edu.vn (V.-K. Tran), nguyenml@jaist.ac.jp (L.-M. Nguyen).

https://doi.org/10.1016/j.csl.2020.101120
0885-2308/

from the source. Hence, simply applying the model trained on the source domain may hurt the performance in the target domain. Furthermore, there is often little or no labeled data in the target domain, which are insufficient to construct a reliable model. Therefore, we mainly aim at achieving good performance on the target domain by leveraging the source data or adapting model trained on the source domain.

Second, *model designing for low-resource setting* has not been well studied in the NLG literature. The generation models have achieved very good performances irrespective of providing adequate labeled datasets (Wen et al., 2015b; 2015a; Tran et al., 2017; Tran and Nguyen, 2017a). For low-resource scenario, one can think about transfer learning which transfers learned representations across domains to improve training on new unseen domains (Dethlefs, 2017), multi-task learning which can be used to transfer dialogue knowledge across different users by sharing training dialogues (Mo et al., 2017), transfer knowledge on multi-lingual data (Mathur et al., 2018) or conversational skills learning via separating out domain-independent dimensions (Keizer and Rieser, 2018), and unsupervised learning (Huang et al., 2018).

Statistical approaches to multi-domain in SDS system have shown promising results in how to efficiently reuse of data in a domain-scalable framework (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 individual domains. Similarly, a approach based on a Bayesian committee machine that used a hierarchical structure to train generic dialogue policies, which can then be refined when the in-domain data is available (Gašić et al., 2017). Recurrent Neural Networks (RNNs) based methods recently have shown improving results in tackling the domain adaptation issue (Chen et al., 2015; Shi et al., 2015; Wen et al., 2016a; 2016b). Wen et al. (2016a) introduced a domain adaptation procedure in which a model is first trained on counterfeited data synthesized from an out-of-domain dataset, and then fine tuned on a small set of in-domain responses with a discriminative objective function. 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., 2015a; Miao et al., 2016; Purushotham et al., 2017; Mnih and Gregor, 2014), dialogue system (Wen et al., 2017; Serban et al., 2017).

This paper present two approaches dealing with the problem of low-resource setting data, in which we first propose an adversarial training procedure to train multi-domain, variational generator via multiple adaptation step which enables the generator to learn more efficiently when in-domain data is in short supply, and second propose a combination of two VAEs, which enables the variational-based generator to learn more efficiently in low-resource setting data.

In summary, we make the following contributions:

- We propose a variational-based NLG framework which benefits the generator to quickly adapt to new, unseen domain irrespective of scarce target resources;
- For domain adaptation, 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, which are integrated into a unifying variational domain adaptation architecture that performs acceptably well in a new, unseen domain by using a limited amount of target data;
- For low-resource model designing, we propose a dual latent variable model which benefits the generator to not only outperform the previous methods when there is a sufficient training data, but also perform acceptably well irrespective of scarce in-domain resources;
- We investigate the effectiveness of the proposed architecture in various scenarios, including domain adaptation, scratch, unsupervised, and semi-supervised training with different amount of training dataset.

The paper is organized as follows. Following a review of related work in Section 2, Section 5 describes in detail a Variational Neural Language Generator framework. While Section 6 presents an Adversarial Variational NLG (VDANLG) for domain adaptation, Section 7 presents a dual variational model for low resource setting in-domain data. The experiments are descried in Section 8, and Section 9 demonstrates the results and analyses. We present our conclusion and future work in Section 10.

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. We thus are interested in improving generating performance on a target domain by using knowledge obtained when generating another related task in source 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, while Mairesse et al. (2010a) have also proposed a phrase-based statistical generator using graphical models and active learning, Cuayáhuitl et al. (2014) proposed an approach to statistical surface realization from unlabeled data through automatic semantic slot labeling.

Domain adaptation for Neural Network based Language Modeling (RNNLMs) has also been studying. While (Shi et al., 2015) proposed three different types of curriculum learning for RNNLMs adaptation, Chen et al. (2015) introduced genre and topic based RNNLM adaptation techniques which were investigated for a multi-genre broadcast transcription task. Consequently, Gangireddy et al. (2016) investigated supervised and unsupervised discriminative adaptation of RNNLMs in a broadcast

transcription task to target domains expounded by either genre or show, whereas Wen et al. (2016a) proposed a procedure to train multi-domain, RNN language generators via data counterfeiting and discriminative training.

For specific domain SDSs, NLG plays an important role in the systems since its task is mainly to convert a meaning representation, *i.e*. dialogue act, into natural language utterances. RNN-based generators have shown improving results in tackling the NLG problem in task oriented-dialogue systems with a varied proposed methods, such as HLSTM (Wen et al., 2015a), SCLSTM (Wen et al., 2015b), and (Tran and Nguyen, 2017b), or especially RNN-based Encoder-Decoder models integrating with attention mechanism, such as Enc-Dec (Wen et al., 2016b), ARoA (Tran et al., 2017), and RALSTM (Tran and Nguyen, 2017a). Such these methods have proved to work well only when providing a sufficient in-domain data since the small training data may harm the model performance. As a result, a question still remains as how to build a generator that can work acceptably well on a small training dataset.

Neural variational framework for generative models of text have been studied extensively. 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), while Zhang et al. (2016) introduced a variational neural machine translation that incorporated a continuous latent variable to model underlying semantics of sentence pairs. Whereas Bowman et al. (2015a) presented a variational autoencoder for unsupervised generative language model, Zhang et al. (2017) proposed a seq2seq purely convolutional and deconvolutional autoencoder solving the exposure-bias problem (Bengio et al., 2015). More recently, Tseng et al. (2018) proposed an SCVAE which integrates the variational auto-encoder into a semantically conditioned for natural language generation.

Adversarial adaptation methods have shown promising improvement in many machine learning applications despite the presence of domain shift or dataset bias. These methods reduce the difference between the training and testing 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. Zhao et al. (2017) proposed a new generalization bound for domain adaptation using adversarial neural networks when there are multiple source domains with annotated samples and one target domain with unannotated samples. 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. NLG Problem decomposition

This section provides a background for most of experiments in this paper, including some task definitions, pre- and post-processing, datasets, evaluation metrics.

3.1. Input meaning representation and training examples

As mentioned, NLG task in SDSs is to convert a meaning representation, yielded by the dialogue manager, into natural language sentences. The meaning representation conveys information of "What to say?" which is represented as a dialogue act (Young et al., 2010). Dialogue act is a combination of an act type and a list of slot-value pairs. Table 1 shows training example pairs of DA-utterance in various NLG domains.

3.2. Datasets

We assessed the proposed models on four different original NLG domains: finding a restaurant, finding a hotel, buying a laptop, and buying a television. All these datasets were released by Wen et al. (2016a). The Restaurant and Hotel were collected in Wen et al. (2015b), while the Laptop and TV datasets released by Wen et al. (2016a). The both latter datasets have a much larger input space but only one training example for each DA, which means the system must learn partial realization of concepts and be able to recombine and apply them to unseen DAs. This also implies that the NLG tasks for the Laptop and TV domains become much harder.

Table 1
Examples of the dialogue act and its corresponding utterance in Hotel, Restaurant, TV, and Laptop domains.

Hotel DA	inform_count(type='*hotel*'; count='*16*'; dogs_allowed='*no*'; near='*dont_care*')
Utterance	There are *16 hotels* that *dogs are not allowed* if you *do not care where it is near to*
Restaurant DA	inform(name='*Ananda Fuara*'; pricerange='*expensive*'; goodformeal='*lunch*')
Utterance	*Ananda Fuara* is a nice place, it is in the *expensive price range* and it is *good for lunch*.
Tv DA	inform_no_match(type='*television*'; hasusbport='*false*'; pricerange='*cheap*')
Utterance	There are no *televisions* which *do not have any usb ports* and in the *cheap price range*.
Laptop DA	recommend(name='*Tecra 89*'; type='*laptop*'; platform='*windows 7*'; dimension='*25.4 inch*')
Utterance	*Tecra 89* is a nice *laptop*. It operates *on windows 7* and its dimensions are *25.4 inch*.

The **Counterfeit** datasets (Wen et al., 2016a) were released by synthesizing Target domain data from Source domain data in order to share realizations between similar slot-value pairs. The datasets are described in detail in Section 3.5 where we assume that we do not have training and validation data from the Target domain yet, except for the test data.

The **Union** datasets were also created by pulling individual datasets together. For example, a [L+T] union dataset were built by merging Laptop and Tv domain data together.

The dataset statistics is shown in Table 2. We also demonstrate the differences of word-level distribution using word clouds in Fig. 1.

3.3. Delexicalization

The number of possible values for a DA slot is theoretically unlimited. This leads the generators to a sparsity problem since there are some slot values which occur only once or even never occur in the training dataset. Delexicalization, which is a pre-process of replacing some slot values with special tokens, brings benefits on reducing data sparsity and improving generalization to unseen slot values since the models only work with delexicalized tokens. Note that the *binary* slots, such as *true/false* or *yes/no* slots, and slots that take *dont_care* cannot be delexicalized since their values cannot exactly match in the training corpus. Table 3 shows some examples of the delexicalization step.

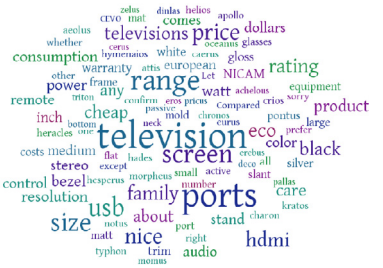
3.4. Lexicalization

Lexicalization procedure in the sentence planning stage is to decide what particular words should be used to express the content. For example, the actual adjectives, adverbs, nouns and verbs to occur in the text are selected from a lexicon. In this study,

Table 2
Dataset statistics.

	Hotel	Restaurant	TV	Laptop
# train	3223	3114	4221	7944
# **10%** train	322	311	422	794
# **30%** train	966	933	1266	2382
# validation	1075	1039	1407	2649
# **10%** validation	107	103	140	264
# **30%** validation	321	309	420	792
# test	1075	1039	1407	2649
# distinct DAs	164	248	7035	13,242
# DA types	8	8	14	14
# slots	12	12	15	19

(a) Laptop domain.



(b) TV domain.

(c) Restaurant domain.

(d) Hotel domain.

Fig. 1. Word clouds for testing set of the four original domains, in which font size indicates the frequency of words.

Table 3
Delexicalization examples.

Hotel DA	inform_only_match(name = '*Nob Hill Motor In*'; dogs_allowed = '*no*'; area = '*Nob hill*'; has_internet = '*yes*')
Reference	The *Nob Hill Motor Inn* is the only hotel in the *Nob hill* area that *has internet* and *does not allow dogs*.
Delexicalized Utterance	The *SLOT_NAME* is the only hotel in the *SLOT_AREA* area that *has internet* and *does not allow dogs*.
Laptop DA	recommend(name='*Satellite Dinlas 18*'; type='*laptop*'; processor='*Intel Celeron*'; is_for_business_computing='*true*'; batteryrating='*standard*')
Reference	The *Satellite Dinlas 18* is a great *laptop for business* with a *standard* battery and an *Intel Celeron* processor
Delexicalized Utterance	The *SLOT_NAME* is a great *SLOT_TYPE* for business with a *SLOT_BATTERYRATING* battery and an *SLOT_PROCESSOR* processor

lexicalization is a post-process of replacing delexicalized tokens with their values to form the final utterances, in which with different slot values we obtain different outputs. Table 4 shows examples of the lexicalization process.

3.5. Counterfeit data

In order to construct an unsupervised domain adaptation scenario in which we do not need any labeled instances from target domain, we synthesized new pseudo training and validating target data from the source data. The procedure is described as follows (see Table 6 for an example):

- Categorize both source and target slots, according to their functional type, into three classes: *requestable, informable*, and **binary** (see Table 5).
- Delexicalize all available slots and their corresponding values.
- Replace each slot **s** in a source instance $(d_i, u_i) \in \mathcal{S}$ with a randomly new slot $\mathbf{s}'$ that is in both the target ontology and the functional type of **s**, yielding a new pseudo instance $(\hat{d}_i, \hat{u}_i) \in \mathcal{T}$ in the target domain.
- Train a generator on the counterfeit data $(\hat{d}_i, \hat{u}_i) \in \mathcal{T}$. Then refine model parameters on real in-domain data. This method allows the model to share realizations among slot-value pairs which have similar functional types.

3.6. Unaligned training data

All four original NLG datasets and their variants used in this study contain *unaligned* training pairs of a dialogue act and corresponding utterance. Our proposed generators in Sections 5, 6, and 7 can *jointly* train both sentence planning and surface realization to convert a MR into natural language utterances. Thus, there is no longer need to explicitly separate training data alignment (Mairesse et al., 2010b; Konstas and Lapata, 2013) which requires domain specific constraints and explicit feature engineering. Examples in Tables 1, 3 and 4 show that correspondences between a DA and words or phrases in its output utterance are not always matched.

Table 4
Lexicalization examples.

Restaurant DA	inform(name='*Connections SF*'; price_range='*pricey*')
Delexicalized Utterance	*SLOT_NAME* is a nice restaurant and it is in the *SLOT_PRICERANGE* price range.
Lexicalized Utterance	*Connections SF* is a nice restaurant and it is in the *pricey* price range.
Hotel DA	inform(name='*Laurel Inn*'; price_range='*moderate*')
Delexicalized Utterance	*SLOT_NAME* is a nice hotel in the *SLOT_PRICERANGE* price range.
Lexicalized Utterance	*Laurel Inn* is a nice hotel in the *moderate* price range.

Table 5
Datasets Ontology.

	Laptop	Television
Act Type	inform*, inform_only_match*, goodbye*, select*, inform_no_match*, inform_count*, request*, request_more*, recommend*, confirm*, inform_all, inform_no_info, compare, suggest	
Requestable Slots	name*, type*, price*, warranty, dimension, battery, design, utility, weight, platform, memory, drive, processor	name*, type*, price*, power_consumption, resolution, accessories, color, audio, screen_size, *family*
Informable Slots	*price_range**, *drive_range, weight_range, family, battery_rating,* ***is_for_business***	*price_range**, *screen_size_range, eco_rating, hdmi_port,* ***has_usb_port***

* = overlap with **Restaurant** and **Hotel** domains, *italic* = slots can take *don't care* value, ***bold*** = binary slots.

Table 6
Counterfeiting procedure. Slots and values are delexicalized. Slots and values that are not in the target domain are then replaced during counteifeiting process (shown in red colour). **I** and **R** prefixes are slot functional types: **I** for Informable and **R** for Requestable.

Source Instance :
TV (Source) DA: inform(name = '*Dinlas 26*'; type = '*television*'; **hasusbport** = '**true**'; powerconsumption = '*32 W*'"; *pricerange* = '*cheap*')
TV (Source) Utterance: *Dinlas 26* is a *television* which **has usb ports**, has *32 W power consumption*, and is in the *cheap price range*.
⇒**Delexicalized TV Utterance**
< *R-NAME-Value* > is a < *R-TYPE-Value* > which **has usb ports**, has < *R-POWERCONSUMPTION-Value* > < *R-POWERCONSUMPTION-Slot* >, and is in the < *I-PRICERANGE-Value* > < *I-PRICERANGE-Slot* >.
⇒**Counterfeiting**
< *R-NAME-Value* > is a < *R-TYPE-Value* > which **is for business computing**, has < *R-BATTERY-Value* > < *R-BATTERY-Slot* >, and is in the < *I-PRICERANGE-Value* > < *I-PRICERANGE-Slot* >.
⇒**A Possible Utterance in Laptop (Target) Domain**
Portege Zelus 80 is a nice *laptop* which **is for business computing**, has a *7.5 h battery*, and is in the *expensive price range*.
Data Counterfeiting Instance: From TV domain to Laptop domain (T2L)
T2L DA: inform(name = "*Dinlas 26*"; type = "*television*"; **isforbusinesscomputing** = "**true**"; battery = "*32 W*"'; *pricerange* = "*cheap*')
T2L Utterance: *Dinlas 26* is a *television* which **has usb ports**, has *32 W power consumption*, and is in the *cheap price range*.

Table 7
Slot error rate (ERR) examples. Errors are marked in colors, such as [missing] and redundant information. [OK] denotes successful generation.

Hotel DA	inform_only_match(name = "*Red Victorian*" ; accepts_credit_cards = "*yes*" ; near = "*Haight*" ; has_internet = "*yes*')
Reference	The *Red Victorian* in the *Haight area* are the only hotel that *accepts credit cards* and *has internet*.
Output A	*Red Victorian* is the only hotel that *allows credit cards near Haight* and *allows internet*. [OK]
Output B	*Red Victorian* is the only hotel that *allows credit cards* and allows credit cards *near Haight* and *allows internet*.
Output C	*Red Victorian* is the only hotel that *nears Haight* and *allows internet*. [allows credit cards]
Output D	*Red Victorian* is the only hotel that *allows credit cards* and allows credit cards and *has internet*. [near Haight]
	Number of total slots in the Hotel domain **N** = 12 (see Table 2)
Output A	ERR = (0+0)/12 = 0.0
Output B	ERR = (0+1)/12 = 0.083
Output C	ERR = (1+0)/12 = 0.083
Output D	ERR = (1+1)/12 = 0.167

4. Evaluation metrics

4.1. BLEU

The Bilingual Evaluation Understudy (BLEU) (Papineni et al., 2002) is often used for comparing a candidate generation of text to one or more reference generations, which is the most frequently used metric for evaluating a generated sentence to a reference sentence. Specifically, the task is to compare *n*-grams of the candidate responses with the *n*-grams of the human-labeled reference and count the number of matches which are position-independent. The more the matches, the better the candidate response is. This study used the cumulative *4*-gram BLEU score (also called BLEU-*4*) for the objective evaluation.

4.2. Slot error rate

The slot error rate ERR (Wen et al., 2015b), which is the number of generated slots that is either redundant or missing, and is computed by:

$$ERR = (\mathbf{s}_m + \mathbf{s}_r)/\mathbf{N} \quad (1)$$

where $\mathbf{s}_m$ and $\mathbf{s}_r$ are the number of missing and redundant slots in a generated utterance, respectively. $\mathbf{N}$ is the total number of slots in given dialogue acts, such as $\mathbf{N} = 12$ for Hotel domain (see Table 7). In some cases when we train adaptation models across domains, we simply set $\mathbf{N} = 42$ is the total number of distinct slots in all four domains. In the decoding phase, for each DA we over-generated 20 candidate sentences and selected the top $\mathbf{k} = 5$ realizations after re-ranking. The slot error rates were computed by averaging slot errors over each of the top $\mathbf{k} = 5$ realizations in the entire corpus. Note that, the slot error rate cannot deal with *dont_care* and *none* values in a given dialogue act. Table 7 demonstrates how to compute the ERR score with some examples. In this study, we adopted code from an NLG toolkit[1] to compute the two metrics BLEU and slot error rate ERR.

[1] https://github.com/shawnwun/RNNLG

5. VNLG - Variational Neural language generator

In this section, we first present a brief introduction about the variational autoencoder in Section 5.1. Section 5.2 provides in detail how to integrate variational autoencoders into an encoder-decoder based generator (Tran and Nguyen, 2017a).

5.1. VAE-Variational Autoencoder

The VAE Kingma and Welling (2013) is a generative model which is mainly based on a standard autoencoder. It introduces a latent variable z, designed to capture the variations in the observed variables $\mathbf{x}$, then the joint distribution is formulated as follows:

$$p(\mathbf{x}, z) = p_\theta(\mathbf{x}|z)p(z) \tag{2}$$

where: θ is the generative model parameters, $p(z)$ is the prior distribution of the latent variable z, i.e., Gaussian distribution, $p(\mathbf{x}|z)$ is the conditional distribution and typically parameterizes via a non-linear deep neural network. However, the posterior inference $p(z|\mathbf{x})$ is intractable and VAE adopts two techniques in order to address this problem: variational neural inference and reparameterization.

Variational neural inference utilizes a neural network to approximate the posterior distribution of latent variable z and formulated as follows:

$$q_\phi(z|\mathbf{x}) = \mathcal{N}\Big(\mu(x), \sigma^2(\mathbf{x})\Big) \tag{3}$$

where: mean $\mu(\mathbf{x})$ and variance $\sigma^2(\mathbf{x})$ are both function of $\mathbf{x}$ parameterized by neural networks.

Reparameterization instead of using the standard sampling method, which reparameterizes z as a function of μ and σ with a standard Gaussian noise variable ϵ and computed as follows:

$$z = \mu + \sigma \odot \epsilon \tag{4}$$

VAE employs an objective function which encourages the model to keep the posterior distribution of z close to its prior distribution, which enables the use of the lower bound. The objective function is formed as follows:

$$\mathcal{L}_{VAE}(\theta, \phi, \mathbf{x}) = -KL\Big(q_\phi(z|\mathbf{x}) \parallel p_\theta(z)\Big) + \mathbb{E}_{q_\phi(z|\mathbf{x})}[\log p_\theta(\mathbf{x}|z)] \leq \log p(\mathbf{x}) \tag{5}$$

where $KL(Q||P)$ is the Kullback-Leibler divergence between Q and P. Maximizing the objective function is equivalent to maximize the reconstruction likelihood of observable variable $\mathbf{x}$ and minimizing the KL divergence between the approximated posterior and the prior distribution of latent variable z.

5.2. VNLG-Variational Natural language generator

Drawing inspiration from VAE model (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 $(\mathbf{d}, \mathbf{u})$, we explicitly model the space together with variable $\mathbf{d}$ to guide the generation process, *i.e.* $p(\mathbf{u}|z, \mathbf{d})$. Thus, the original conditional probability is reformulated as follows:

$$p(\mathbf{u}|\mathbf{d}) = \int_z p(\mathbf{u}, z|\mathbf{d})\mathbf{d}_z = \int_z p(\mathbf{u}|z, \mathbf{d})p(z|\mathbf{d})\mathbf{d}_z \tag{6}$$

This latent variable enables us to model the underlying semantic space as a global signal for generation. However, the incorporating of latent variable into the probabilistic model arises two difficulties in *(i)* modeling the intractable posterior inference $p(z|\mathbf{d}, \mathbf{u})$ and *(ii)* whether or not the latent variables z can be modeled effectively in case of low-resource setting data.

To address the difficulties, we propose an encoder-decoder based variational model to natural language generation (VNLG) by integrating a variational autoencoder (Kingma and Welling, 2013) into an encoder-decoder generator (Tran and Nguyen, 2017a). Fig. 4-*(a)* shows a graphical model of VNLG. We then employ deep neural networks to approximate the prior $p(z|\mathbf{d})$, true posterior $p(z|\mathbf{d}, \mathbf{u})$, and decoder $p(\mathbf{u}|z, \mathbf{d})$. To tackle the first issue, the intractable posterior is approximated from both the DA and utterance information $q_\phi(z|\mathbf{d}, \mathbf{u})$ under the above assumption. In contrast, the prior is modeled to condition on the DA only $p_\theta(z|\mathbf{d})$ due to the fact that the DA and utterance of a training pair usually share the same semantic information, i.e., a given DA *inform*(name='*ABC*'; area='*XYZ*') contains key information of the corresponding utterance "The hotel *ABC* is in *XYZ* area". The underlying semantic space with having more information encoded from both the prior and the posterior provides the generator a potential solution to tackle the second issue in the *domain adaptation* and *scratch* training scenarios. Lastly, in generative process, given an observation DA $\mathbf{d}$ the output $\mathbf{u}$ is generated by the decoder network $p_\theta(\mathbf{u}|z, \mathbf{d})$ under the guidance of the global signal z which is drawn from the prior distribution $p_\theta(z|\mathbf{d})$. According to (Kingma and Welling, 2013; Sohn et al., 2015), the variational lower bound can be recomputed as:

$$\mathcal{L}_{VAE}(\theta, \phi, \mathbf{d}, \mathbf{u}) = -KL\Big(q_\phi(z|\mathbf{d}, \mathbf{u}) \parallel p_\theta(z|\mathbf{d})\Big) + \mathbb{E}_{q_\phi(z|\mathbf{d}, \mathbf{u})}[\log p_\theta(\mathbf{u}|z, \mathbf{d})] \tag{7}$$

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

The variational architecture for natural language generation is demonstrated in Fig. 2, in which a variational inference is integrated into an encoder-decoder based natural language generator (Tran and Nguyen, 2017a). The variational generation architecture comprises three components: Variational Encoder Network (Section 5.2.1), Variational Inference Network (Section 5.2.2), and Variational Decoder Network (Section 5.2.3).

5.2.1. Variational encoder network

The variational encoder network consists of two networks: (*i*) a 1-layer, Bidirectional LSTM (BiLSTM) encoding the sequence of slot-value pairs $\{\mathbf{sv}_i\}_{i=1}^{T_{DA}}$ in a given Dialogue Act; and (*ii*) a shared RNN/CNN Encoder encoding the given input utterance $\mathbf{u}$. The input sequence $\mathbf{u}$ of length T_U (padded where necessary) represented as $\mathbf{U} \in \mathbb{R}^{d \times T_U}$ by concatenating its word embedding $\mathbf{U}_t \in \mathbf{E}[\mathbf{u}_t]$, where $\mathbf{E} \in \mathbb{R}^{d \times |\mathcal{V}|}$, d, $|\mathcal{V}|$ are embedding and vocabulary sizes, respectively. All columns of $\mathbf{E}$ are normalized to have unit l_2-norm. The encoder, thus, produces both the DA representation and the utterance representation vectors which flow into the inference and decoder networks, and the posterior approximator, respectively.

BiLSTM dialogue act encoder

The BiLSTM consists of forward and backward LSTMs which process the sequence from left-to-right and right-to-left, yielding both forward and backward sequence of hidden states $(\overrightarrow{\mathbf{h}}_1, .., \overrightarrow{\mathbf{h}}_{T_{DA}})$, and $(\overleftarrow{\mathbf{h}}_1, .., \overleftarrow{\mathbf{h}}_{T_{DA}})$, respectively. We finally take the mean-pooling over the BiLSTM hidden vectors to obtain the Dialogue Act representation: $\mathbf{h_D} = \frac{1}{T_{DA}} \sum_i^{T_{DA}} \mathbf{h}_i$, where $\mathbf{h}_i = \overrightarrow{\mathbf{h}}_i + \overleftarrow{\mathbf{h}}_i$.

CNN utterance encoder

We use CNN utterance encoder for constructing a low-resource setting generator which is described in Section 7. The CNN consists of L-1 convolutional layers and a L-th fully-connected layer, which aims at encoding an input utterance $\mathbf{u}$ into a fixed length representation vector $\mathbf{h_U}$. Layer $l \in \{1, .., L\}$ comprises learnable k_l filters. For j-th filter in layer $l = [1, .., L-1]$, a convolutional operation with stride length $s^{(l)}$ applies filter $\mathbf{W}_v^{(j,l)} \in \mathbb{R}^{d \times h}$, where h is convolutional filter size. This produces latent feature map, $\mathbf{v}^{(j,l)} = ReLU(\mathbf{U} * \mathbf{W}_c^{(j,l)} + b^{(j,l)}) \in \mathbb{R}^{(T^{(l)}-h)/s^{(l)}+1}$, where $b^{(j,l)} \in \mathbb{R}^{(T^{(l)}-h)/s^{(l)}+1}$ is bias, and * is the convolutional operator. We finally concatenate the results from k_l filters, results in feature map $\mathbf{V}^{(l)} = [v^{(1,l)}, \ldots, v^{(k_l,l)}] \in \mathbb{R}^{k_l \times [(T^{(l)}-h)/s^{(l)}+1]}$. For each layers $l = [1, .., L-1]$, the length along the spatial dimension is reduced to $T^{(l+1)} = \lfloor (T^{(l)} - h)/s^{(l)} + 1 \rfloor$, where $T^{(l)}$, $s^{(l)}$ are the spatial length and the stride length, respectively, and $\lfloor . \rfloor$ is the floor function. The feature map $\mathbf{V}^{(L-1)}$, at the final layer L, is fed into a fully-connected layer to yield the latent representation $\mathbf{h_U}$ which encapsulates the sentence sub-structure via the whole sentence portrayed by filters $\{W_v^{(j,l)}\}$. We utilize the implementation trick as in Radford et al. (2015), in which we use a convolutional layer with the filter size equals to $T^{(L-1)}$.

In this work, for example, the CNN encoder consists of $L = 3$ layers, which for a sentence of length $T_U = 73$, embedding size $d = 100$, stride length $s = \{2, 2, 2\}$, number of filters $k = \{300, 600, 100\}$ with filter sizes $h = \{5, 5, 16\}$, results in feature maps $\mathbf{V}$ of sizes $\{35 \times 300, 16 \times 600, 1 \times 100\}$, in which the last feature map corresponds to latent representation vector $\mathbf{h_U}$.

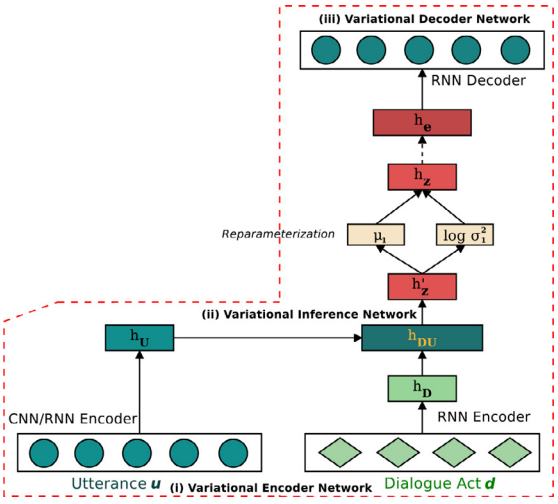


Fig. 2. The Variational NLG architecture. The model consists of three main components: (i) Variational Encoder Network, (ii) Variational Inference Network, and (iii) Variational Decoder Network.

RNN utterance encoder

For constructing a domain-adaptation generator, we utilize RNN Utterance Encoder whose architecture is same as BiLSTM DA encoder in Section 5.2.1. The final representation the corresponding utterance $\{\mathbf{u}_i\}_{i=1}^{T_U}$ is obtained by taking the mean-pooling over the BiLSTM hidden vectors $\mathbf{h}_U = \frac{1}{T_U}\sum_i^{T_U} \mathbf{h}'_i$ where $\mathbf{h}'_i = \overrightarrow{\mathbf{h}'_i} + \overleftarrow{\mathbf{h}'_i}$.

5.2.2. Variational inference network

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

Neural posterior approximator

Modeling the true posterior $p(z|\mathbf{d},\mathbf{u})$ 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:

$$q_\phi(z|\mathbf{d},\mathbf{u}) = \mathcal{N}\Big(z; \mu_1\big(f(\mathbf{h}_D,\mathbf{h}_U)\big), \sigma_1^2\big(f(\mathbf{h}_D,\mathbf{h}_U)\big)I\Big) \tag{8}$$

where mean μ_1 and standard variance σ_1 are outputs of the neural network based on the representations of $\mathbf{h}_D$ and $\mathbf{h}_U$. The function f is a non-linear transformation that project both DA and utterance representations into the latent space:

$$\mathbf{h}'_z = f(\mathbf{h}_D,\mathbf{h}_U) = g(\mathbf{W}_z[\mathbf{h}_D;\mathbf{h}_U] + b_z) \tag{9}$$

where $\mathbf{W}_z \in \mathbb{R}^{d_z \times (d_{\mathbf{h}_D} + d_{\mathbf{h}_U})}$ and $b_z \in \mathbb{R}^{d_z}$ 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_1 and $\log\sigma_1^2$ through linear regression:

$$\mu_1 = \mathbf{W}_{\mu_1}\mathbf{h}'_z + b_{\mu_1},\ \log\sigma_1^2 = \mathbf{W}_{\sigma_1}\mathbf{h}'_z + b_{\sigma_1} \tag{10}$$

where $\mu_1,\ \log\sigma_1^2$ are both d_z dimension vectors.

Neural prior model

We model the prior as follows:

$$p_\theta(z|\mathbf{d}) = \mathcal{N}\Big(z; \mu'_1(\mathbf{d}), \sigma'_1(\mathbf{d})^2 I\Big) \tag{11}$$

where μ'_1 and σ'_1 of the prior are neural models based on DA representation only, which are the same as those of the posterior $q_\phi(z|\mathbf{d},\mathbf{u})$ in Eq. 8 and Eq. 10, except for the absence of $\mathbf{h}_U$. 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:

$$\mathbf{h}_z = \mu_1 + \sigma_1 \odot \epsilon, \epsilon \tilde{} \mathcal{N}(0, I) \tag{12}$$

In addition, we set $\mathbf{h}_z$ to be the mean of the prior $p_\theta(z|\mathbf{d})$, *i.e.*, μ'_1, during decoding due to the absence of the utterance $\mathbf{u}$. 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:

$$\mathbf{h}_e = g(\mathbf{W}_e\mathbf{h}_z + b_e) \tag{13}$$

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 9 and Fig. 6).

5.2.3. Variational neural decoder

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

$$p(\mathbf{u}|z,\mathbf{d}) = \prod_{j=1}^{T_U} p(\mathbf{u}_t|\mathbf{u}_{<t}, z, \mathbf{d}) \tag{14}$$

where: $p(\mathbf{u}_t|\mathbf{u}_{<t}, z, \mathbf{d}) = g'(RNN(\mathbf{u}_t, \mathbf{h}_{t-1}, \mathbf{d}_t)$ In this paper, we borrow the $\mathbf{d}_t$ calculation and the computational RNN cell from work (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 Fig. 2-(iii). We modify the cell calculation as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W}_{4d_h,4d_h} \begin{pmatrix} \mathbf{h}_e \\ \mathbf{d}_t \\ \mathbf{h}_{t-1} \\ \mathbf{u}_t \end{pmatrix} \tag{15}$$

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 Inference RALSTM (VI-RALSTM) model with CNN utterance encoder (VIC-RALSTM) or with RNN utterance encoder (VIR-RALSTM) are demonstrated in Fig. 2-(i), (ii), (iii), in which the latent variable affects the hidden representation through the gates. This allows the model can indirectly take advantage of the underlying semantic information from the latent variable z. Furthermore, 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 Section 9.2.3 for details).

6. VDANLG - An adversarial VNLG for domain adaptation

In this Section we propose two novel critics which guide the VNLG-based model to adapt quickly to a new domain, we then propose a novel adversarial training procedure for domain adaptation. Note that we use VIR-RALSTM (see Section 5.2.1) in this setting, resulting in a Variational Domain Adaptation NLG (VDANLG) model is illustrated in Fig. 3.

6.1. Critics

This Section introduces 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.

6.1.1. Text similarity critic

In order to examine 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 $(\mathbf{u}_{(1)}, \mathbf{u}_{(2)})$ as 1-similar or 0-unsimilar text style. The SC model consists of two parts: a shared BiLSTM $\mathbf{h}_Y$ with the Variational Neural Encoder to represent the $\mathbf{u}_{(1)}$ sentence, and a second BiLSTM to encode the $\mathbf{u}_{(2)}$ sentence. The SC model takes input as a pair $(\mathbf{u}_{(1)}, \mathbf{u}_{(2)})$ of ([*target*], *source*), ([*target*], *generated*), and ([*generated*], *source*). Note that we give priority to encoding the $\mathbf{u}_{(1)}$ 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 the model to learn useful representations with limited supervision.

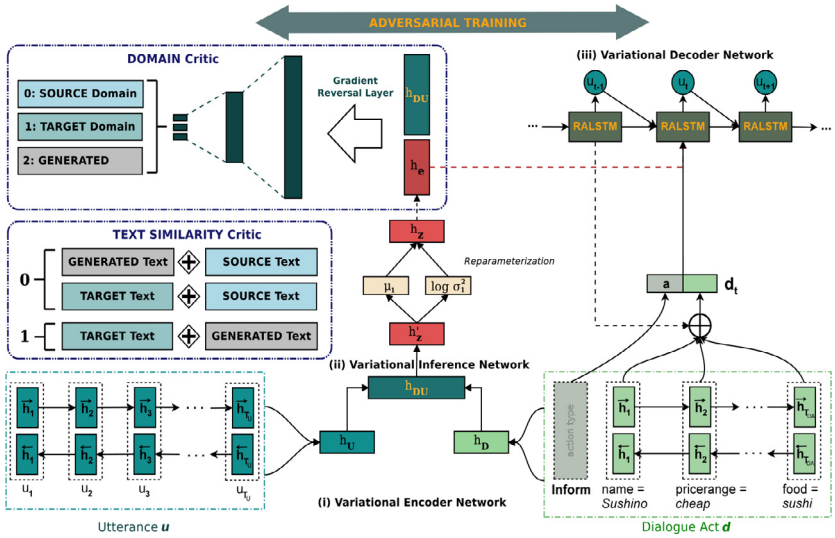


Fig. 3. The Variational Domain Adaptation NLG (VDANLG) architecture. The model consists of two main components: the VIR-RALSTM to generate the sentence, which comprises : (i) Variational Encoder Network, (ii) Variational Infererence Network, and (iii) Variational Decoder Network; and two Critics with an adversarial training procedure to guide the model in domain adaptation, which is composed of a Domain critic and a Text Similarity critic.

6.1.2. Domain critic

In order to learn a model that can generalize well from a source domain to a new target domain, and more specifically, in consideration of the shifts 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 certain 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.

6.2. Training domain adaptation model

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

6.2.1. 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:

$$\begin{aligned}
&\mathcal{L}_s(\psi) = -\sum_{n=1}^{N} \log C_s(l_s^n | \mathbf{u}_{(1)}^n, \mathbf{u}_{(2)}^n, \psi), \\
&l_s^n = \begin{cases} 1-similar & if\ (\mathbf{u}_{(1)}^n, \mathbf{u}_{(2)}^n) \in \mathcal{P}_{sim}, \\ 0-unsimilar & if\ (\mathbf{u}_{(1)}^n, \mathbf{u}_{(2)}^n) \in \mathcal{P}_{unsim}, \end{cases} \mathcal{U}_{\mathcal{G}} = \{\mathbf{u} | \mathbf{u} \sim \mathcal{G}(.|\mathbf{d}_{\mathcal{T}}, .)\}, \mathcal{P}_{sim} = \{\mathbf{u}_{\mathcal{T}}^n, \mathbf{u}_{\mathcal{U}_{\mathcal{G}}}^n\}, \mathcal{P}_{unsim} = (\{\mathbf{u}_{\mathcal{T}}^n, \mathbf{u}_{\mathcal{S}}^n\}, \{\mathbf{u}_{\mathcal{U}_{\mathcal{G}}}^n, \mathbf{u}_{\mathcal{S}}^n\})
\end{aligned} \tag{16}$$

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

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:

$$\mathcal{L}_d(\varphi) = -\sum_{n=1}^{N} \log C_d(l_d^n | \mathbf{d}^n, \mathbf{u}^n, \varphi), l_d^n = \begin{cases} source & if\ (\mathbf{d}^n, \mathbf{u}^n) \in (\mathcal{D}_{\mathcal{S}}, \mathcal{U}_{\mathcal{S}}), \\ target & if\ (\mathbf{d}^n, \mathbf{u}^n) \in (\mathcal{D}_{\mathcal{T}}, \mathcal{U}_{\mathcal{T}}), \\ generated & if\ (\mathbf{d}^n, \mathbf{u}^n) \in (\mathcal{D}_{\mathcal{T}}, \mathcal{U}_{\mathcal{G}}), \end{cases} \tag{17}$$

where φ is the model parameters of DC, and $(\mathcal{D}_S, \mathcal{U}_S)$ and $(\mathcal{D}_T, \mathcal{U}_T)$ are the DA-utterance pairs from source and target domain, respectively; $\mathcal{U}_{\mathcal{G}}$ denotes sentences generated from the current generator $\mathcal{G}$ given target domain dialogue act $\mathbf{d}_{\mathcal{T}}$. Note also that the scalar probability $C_d(target|\mathbf{d}^n, \mathbf{u}^n)$ indicates how likely the DA-utterance pair $(\mathbf{d}^n, \mathbf{u}^n)$ is from the target domain.

6.2.2. Training variational neural language generator

We utilize the Monte Carlo method to approximate the expectation over the posterior in Eq. 7, *i.e.* $\mathbb{E}_{q_\phi(z|\mathbf{d},\mathbf{u})}[.] \simeq \frac{1}{M}\sum_{m=1}^{M} \log p_\theta(\mathbf{u}|\mathbf{d}, \mathbf{h}_z^{(m)})$ where: M is the number of samples. In this study, the joint training objective for a training instance $(\mathbf{d}, \mathbf{u})$ is formulated as follows:

$$\mathcal{L}(\theta, \phi, \mathbf{d}, \mathbf{u}) \simeq -KL\Big(q_\phi(z|\mathbf{d}, \mathbf{u}) \parallel p_\theta(z|\mathbf{d})\Big) + \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T_u} \log p_\theta(\mathbf{u}_t|\mathbf{u}_{<t}, \mathbf{d}, \mathbf{h}_z^{(m)}) \tag{18}$$

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. 18 is differentiable, we can jointly optimize the parameter θ and variational parameter ϕ using standard gradient ascent techniques.

6.2.3. Adversarial training

Our domain adaptation architecture is demonstrated in Fig. 2, 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 $\mathbf{u}$'s. It would prefer a "*good*" generated sentence $\mathbf{u}$ if the values of $C_d(target|\mathbf{d}_{\mathcal{T}}, \mathbf{u})$ and $C_s(1|\mathbf{u}_{\mathcal{T}}, \mathbf{u})$ are large. In contrast, the critics would prefer large values of $C_d(generated|\mathbf{d}_{\mathcal{T}}, \mathbf{u})$ and $C_s(1|\mathbf{u}, \mathbf{u}_{\mathcal{S}})$, which imply the small values of $C_d(target|\mathbf{d}_{\mathcal{T}}, \mathbf{u})$ and $C_s(1|\mathbf{u}_{\mathcal{T}}, \mathbf{u})$. 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 is optimized to minimize its 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.

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 $\mathcal{U}_{\mathcal{G}}$ of K candidate sentences (step 11). We then choose top $\mathbf{k}$ best sentences in the $\mathcal{U}_{\mathcal{G}}$ set (step 12) after re-ranking to measure how

Algorithm 1. Adversarial Training Procedure.

Require: generator $\mathcal{G}$, domain critic C_d, text similarity critic C_s, generated sentence $\mathcal{U}_{\mathcal{G}} = \emptyset$;
Input: DA-utterance pairs of source $(\mathcal{D}_{\mathcal{S}}, \mathcal{U}_{\mathcal{S}})$, target $(\mathcal{D}_{\mathcal{T}}, \mathcal{U}_{\mathcal{T}})$;
Pretrain $\mathcal{G}$ on $(\mathcal{D}_{\mathcal{S}}, \mathcal{U}_{\mathcal{S}})$ using VIR-RALSTM (see Section 5.2.1);
while Θ *has not converged* **do**
  for i = *0, .., $N_{\mathcal{T}}$* **do**
    Sample $(\mathbf{d}_{\mathcal{S}}, \mathbf{u}_{\mathcal{S}})$ from source domain;
    (D_1)-Compute $g_d = \nabla_{\varphi} \mathcal{L}_d(\varphi)$ using Eq. 17 for $(\mathbf{d}_{\mathcal{S}}, \mathbf{u}_{\mathcal{S}})$ and $(\mathbf{d}_{\mathcal{T}}, \mathbf{u}_{\mathcal{T}})$;
    (D_2)-Adam update of φ for C_d using g_d;
    (G_1)-Compute $g_{\mathcal{G}} = \{\nabla_{\theta} \mathcal{L}(\theta, \phi), \nabla_{\phi} \mathcal{L}(\theta, \phi)\}$ using Eq. 18
    (G_2)-Adam update of θ, ϕ for $\mathcal{G}$ using $g_{\mathcal{G}}$
    (S_1)-Compute $g_s = \nabla_{\psi} \mathcal{L}_s(\psi)$ using Eq. 16 for $(\mathbf{u}_{\mathcal{T}}, \mathbf{u}_{\mathcal{S}})$;
    (S_2)-Adam update of ψ for C_s using g_s;
    $\mathcal{U}_{\mathcal{G}} \leftarrow \{\mathbf{u}_{\bar{k}}\}_{\bar{k}=1}^{K}$, where $\mathbf{u}_{\bar{k}} \sim \mathcal{G}(.|\mathbf{d}_{\mathcal{T}}^{(i)}, .)$;
    Choose top $\boldsymbol{k}$ best sentences of $\mathcal{U}_{\mathcal{G}}$;
    for j = *1,..,$\boldsymbol{k}$* **do**
      (D_1), (D_2) steps for C_d with $(\mathbf{d}_{\mathcal{T}}, \mathbf{u}_{\mathcal{G}}^{(j)})$;
      (S_1), (S_2) steps for C_s with $(\mathbf{u}_{\mathcal{G}}^{(j)}, \mathbf{u}_{\mathcal{S}})$ and $(\mathbf{u}_{\mathcal{T}}, \mathbf{u}_{\mathcal{G}}^{(j)})$;
    end
  end
end

"*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 $\mathcal{U}_{\mathcal{G}}$ by penalizing the generated sentences which have redundant or missing slots. This helps the model to produce the utterances with lower ERR score (see Table 9).

7. DualVAE - A dual variational model for low-Resource setting data

Starting from a Varitional neural language generator with a CNN utterance encoder (VIC-RALSTM) in Section 5.2.1, we present an effective way to construct a dual Variational model which consists of two VAEs and enables the variational-based generator to learn more efficiently when the training data is in short supply. In more details, we integrate a variational inference into an encoder-decoder generator and introduce a novel auxiliary auto-encoding with an effective training procedure. Fig. 4 shows a graphical model of the proposed dual variational model. The following Section 7.1 presents a second auxiliary VAE model which is a Variational CNN-DCNN model as shown in the left side of Fig. 5. Section 7.2 then proposes a novel training procedure to effectively leverage knowledge from a small amount of training data.

7.1. Variational CNN-DCNN model

This standard VAE model (left side in Fig. 5) acts as an auxiliary auto-encoding for utterance (used at training time) to the VNLG generator. This VAE model consists of two components: a shared CNN Utterance Encoder model with the Variational Language Generator, and a DCNN Utterance Decoder model. After having the vector representation $\mathbf{h}_{\mathbf{U}}$, we apply another linear regression to obtain the distribution parameter μ_2 and $\log\sigma_2^2$ as follows:

$$\mu_2 = \mathbf{W}_{\mu_2}\mathbf{h}_{\mathbf{U}} + b_{\mu_2},\ \log\sigma_2^2 = \mathbf{W}_{\sigma_2}\mathbf{h}_{\mathbf{U}} + b_{\sigma_2} \tag{19}$$

where: μ_2, $\log\sigma_2^2$ are also both d_z dimension vectors. We also obtain a representation of the latent variable z by re-parameterizing it as follows:

$$\mathbf{h}_{zu} = \mu_2 + \sigma_2 \odot \epsilon, \epsilon \tilde{} \mathcal{N}(0, I) \tag{20}$$

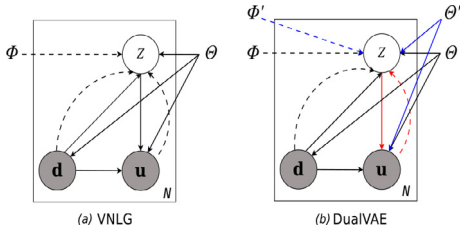


Fig. 4. Illustration of proposed variational models as a directed graph. *(a)* VNLG: joint learning both variational parameters ϕ and generative model parameters θ. *(b)* DualVAE: red and blue arrows form a standard VAE (parameterized by ϕ' and θ') as an auxiliary auto-encoding to the VNLG model denoted by red and black arrows. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

In order to integrate the latent variable $\mathbf{h}_{zu}$ into the DCNN Decoder, we use a shared non-linear transformation as in Eq. 13 (dashed black line in Fig. 5):

$$\mathbf{h}_e = g(\mathbf{W}_e \mathbf{h}_{zu} + b_e) \tag{21}$$

7.1.1. DCNN utterance decoder

To decode the latent representation, $\mathbf{h}_e$, back to the source text, we use the deconvolutional network with stride, also known as transposed convolutional layers. As a minoring the convolutional steps, the spatial dimension first is expanded to match those of the $(L-1)$-th convolutional layer, then progressively widened as $T^{(l+1)} = (T^{(l)} - 1) * s^{(l)} + h$ for $l = 1.L$, which corresponds to the input layer of the CNN utterance encoder. The output of the L-th deconvolutional layer aims to recontruct the word embedding matrix denoted as $\hat{\mathbf{U}}$ whose columns are normalized to have unit l_2-norm as well as word embedding matrix $\mathbf{E}$. The probability of $\hat{\mathbf{u}}_t$ to be word s is computed as follows:

$$p(\hat{\mathbf{u}}_t = s) = \frac{exp\{\tau^{-1} cos(\hat{\mathbf{u}}_t, \mathbf{E}[s])\}}{\sum_{s' \in \mathcal{V}} exp\{\tau^{-1} cos(\hat{\mathbf{u}}_t, \mathbf{E}[s'])\}} \tag{22}$$

where: $cos(x, y)$ is the cosine similarity between two vectors x and y, $\mathcal{V}$ is the word vocabulary, $\mathbf{E}[s]$ denotes the column of word embedding $\mathbf{E}$ corresponding to word s. Temperature parameter τ is set to be 0.01 to control the sparsity of the resulting probabilities.

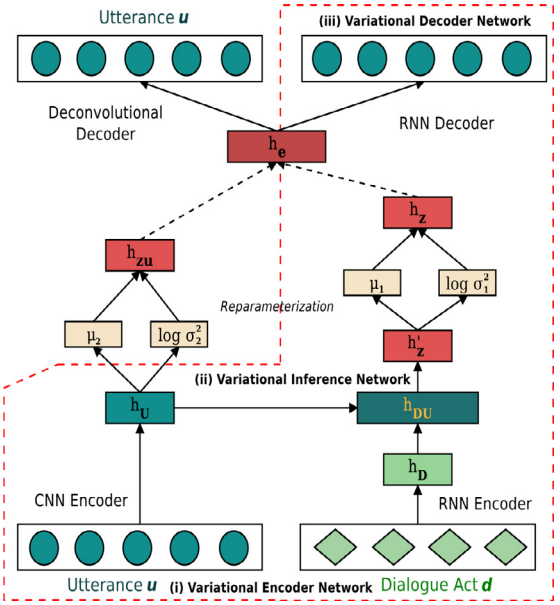


Fig. 5. The Dual Variational Model consists of two VAE models: (I) the Variational Natural Language Generator (VIC-RALSTM) in the dashed red box to generate utterances, which comprises: (i) Variational Encoder Network, (ii) Variational Inference Network, and (iii) Variational Decoder Network; and (II) the Variational CNN-DCNN Model (left side) which is an auxiliary auto-encoding model (left side) and composed of a CNN Encoder and a Deconvolutional Decoder. The CNN Encoder for utterance encoding is shared between the two VAEs. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

The resulting model named DualVAE by incorporating the Variational NLG (VIC-RALSTM) with the Variational CNN-DCNN model and depicted in Fig. 5.

7.2. Training dual latent variable model

7.2.1. Training variational language generator

Similar to Section 6.2.2 on training a variational NLG, the joint training objective for a training instance pair $(\mathbf{d}, \mathbf{u})$ is formulated as follows:

$$\mathcal{L}_{VIC\text{–}RALSTM} = \mathcal{L}(\theta, \phi, \mathbf{d}, \mathbf{u}) \tag{23}$$

where $\mathcal{L}(\theta, \phi, \mathbf{d}, \mathbf{u})$ is computed as in Eq. 18.

7.2.2. Training variational CNN-DCNN model

The objective function of the Varialtional CNN-DCNN model is the standard VAE lower bound (Kingma and Welling, 2013) to be maximized as follows:

$$\mathcal{L}_{CNN\text{–}DCNN} = \mathcal{L}(\theta', \phi', \mathbf{u}) = -KL\Big(q_{\phi'}(z|\mathbf{u}) \parallel p_{\theta'}(z)\Big) + \mathbb{E}_{q_{\phi'}(z|\mathbf{u})}[\log p_{\theta'}(\mathbf{u}|z)] \leq \log p(\mathbf{u}) \tag{24}$$

where θ' and ϕ' denote decoder and encoder parameters, respectively. Intuitively, maximizing the objective function is equivalent to maximize the reconstruction likelihood of observable variable $\mathbf{u}$ and minimizing the KL divergence between the approximated posterior and the prior distribution of latent variable z. During training, we also consider a denoising autoencoder where we slightly modify the input by swapping some arbitrary word pairs.

7.2.3. Joint training dual VAE model

To allow the model explore and balance maximizing the variational lower bound between the CNN-DCNN model and VIC-RALSTM model, an objective is joint dual training as follows:

$$\mathcal{L}_{DualVAE} = \mathcal{L}_{VIC\text{–}RALSTM} + \alpha \mathcal{L}_{CNN\text{–}DCNN} \tag{25}$$

where α controls the relative weight between two variational losses. During training, we anneal the value of α from 1 to 0, so that the dual latent variable learned can gradually focus less on reconstruction objective of the CNN-DCNN model, only retain those features that are useful for the generation objective.

7.2.4. Joint cross training dual VAE model

To allow the dual VAE model explore and encode useful information of the dialogue act into the latent variable, we further take a cross training between two VAEs by simply replacing the RALSTM Decoder of the VIC-RALSTM with the DCNN Utterance Decoder, and its objective training as:

$$\mathcal{L}_{VIC\text{–}DCNN} = \mathcal{L}(\theta', \phi, \mathbf{d}, \mathbf{u}) \simeq -KL\Big(q_{\phi}(z|\mathbf{d}, \mathbf{u}) \parallel p_{\theta'}(z|\mathbf{d})\Big) + \mathbb{E}_{q_{\phi}(z|\mathbf{d},\mathbf{u})}[\log p_{\theta'}(\mathbf{u}|z, \mathbf{d})], \tag{26}$$

and a joint cross training objective is employed:

$$\mathcal{L}_{CrossVAE} = \mathcal{L}_{VIC\text{–}RALSTM} + \alpha(\mathcal{L}_{CNN\text{–}DCNN} + \mathcal{L}_{VIC\text{–}DCNN}) \tag{27}$$

8. Experiments

8.1. 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. We used Adam optimizer with the learning rate 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. For the variational inference, we set the latent variable size to be 16 for VDANLG model and 300 for dual VAEs.

8.2. Decoding

The decoding we implemented here is similar to those in work of Wen et al. (2015b), which consists of two phases: (i) over-generation, and (ii) re-ranking. In the first phase, the generator RALSTM decoder, conditioned on both representations of the given DA and the latent variable, uses a beam search (with beam size is set to be 10) to generate a set of 20 candidate responses. The objective cost of the generator, in the re-ranking phase, is calculated to form the re-ranking score R as follows:

$$R = \mathcal{L}(.) + \lambda ERR \tag{28}$$

where: $\mathcal{L}(.)$ is cost of variational generator training, λ is a trade-off constant and is set to be 1000 to severely penalize nonsensical outputs. The slot error rate ERR Wen et al. (2015b), which is the number of slots generated that is either redundant or missing, and is computed as:

$$ERR = (\mathbf{s}_m + \mathbf{s}_r)/\mathbf{N} \tag{29}$$

where: $\mathbf{N}$ is the total number of slots in given dialogue act, and $\mathbf{s}_m$, $\mathbf{s}_r$ is the number of missing and redundant slots, respectively. In the decoding phase, for each DA we over-generated 20 candidate sentences and selected the top $\mathbf{k} = 5$ realizations after re-ranking. The slot error rates were computed by averaging slot errors over each of the top 5 realizations in the entire corpus.

8.3. KL Cost annealing

VAE is hard to train because of the model in most cases converges to a solution with a vanishing small KL term, thus effectively falling back to a conventional language model. Following (Bowman et al., 2015b), we use KL cost annealing strategy to encourage the model to encode meaningful representations into the z latent vector, in which we gradually annealing the KL term from 0 to 1. This helps our model to achieve solutions with non-zero KL term.

8.4. Gradient reversal layer

The 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 during the backpropagation-based training. For configuring the Domain critic (see 6.1) in VDANLG model, 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 as follows:

$$\begin{aligned} p &= float(i)/num\_steps, \\ \lambda_p &= \frac{2}{1 + \exp(-10 * p)} - 1 \end{aligned} \tag{30}$$

where: *num_steps* is a constant which is set to be 8600, p is the training progress. This strategy allows the Domain critic to be less sensitive to noisy signal at the early stages of the training procedure.

8.5. Evaluation metrics and baselines

The generator performances were evaluated using the two metrics: the BLEU and the slot error rate ERR by adopting code from an NLG toolkit.[2] We compared the proposed models against strong baselines which have been recently published as NLG benchmarks of the above datasets.

- Gating-based generators, including SCLSTM model (Wen et al., 2015b) which jointly learns the gating signal and language model by using a semantic reading gate, and HLSTM model (Wen et al., 2015a) which uses a heuristic gate to ensure that all of the attribute-value information was accurately captured during generation.
- Attention-based generators, including Enc-Dec model (Wen et al., 2016b) which applies the attention mechanism to an RNN encoder-decoder, and a hybrid RALSTM model (Tran and Nguyen, 2017a) which is a combination of an attention over the slot-value pairs with gating a control dialogue act vector.

9. Results and analysis

We performed the models in different scenarios as follows:

- *Scratch* training: Models trained from scratch using 10% (*scr10*), 30% (*scr30*), and 100% (*scr100*) amount of in-domain data;
- Domain *adaptation* training: Models pre-trained from scratch using all source domain data, then fine-tuned on the target domain using only 10% amount of the target data.

Overall, both proposed models demonstrate an ability to work well in various scenarios of low-resource setting data. The proposed models further obtained better performance regarding both the evaluation metrics across all domains in all training scenarios. We start investigating the effectiveness of variational integrating in Section 9.1. Sections 9.2 and 9.3 present the results and analyses of domain adaptation models and dual variational models, respectively.

[2] https://github.com/shawnwun/RNNLG .

9.1. Integrating variational inference

We compare the original model RALSTM with its modification by integrating with Variational Inference (VIR-RALSTM and VIC-RALSTM) as demonstrated in Fig. 6 and Table 8. It clearly shows that the model integration not only preserves the power of the original RALSTM on generation task since its performances are very competitive to those of RALSTM (Table 8, *sec*.1), 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% (Fig. 6). Table 8, *sec*.2 further shows the necessity of the integrating in which the Variational RALSTM achieved a significant improvement over the RALSTM in *scr10* scenario where the models trained from *scratch* with only a limited amount of training data (10%). These indicate that the proposed variational method can learn the underlying semantic of the existing DA-utterance pairs, which are especially useful information for low-resource setting.

Furthermore, the VIR-RALSTM model has slightly better results than the VIC-RALSTM when providing sufficient training data, *i.e*., 100%. In contrast, with a limited training data, *i.e*, 10%, the latter model demonstrates a significant improvement compared to previous models in terms of both BLEU and ERR scores by a large margin across all four dataset. In Hotel domain, for example, the VIC-RALSTM model (79.98 BLEU, 8.67% ERR) has better results in comparison to the VIR-RALSTM (73.78 BLEU, 15.43% ERR) and RALSTM (68.55 BLEU, 22.53% ERR). The VIC-RALSTM, the model with CNN utterance encoder, shows obvious sign for constructing a dual latent variable models dealing with the limitation of in-domain data, which are discussed in Section 9.3. The following Section 9.2 provides in detail results and analyses of the VDANLG model in tackling domain adaptation problems.

9.2. Adversarial VNLG for domain adaptation

We compared the Variational Domain Adaptation NLG (see Section 6) against the baselines in various scenarios: *adaptation, scr10, scr100*. Overall, the proposed models trained on *adaptation* scenario not only achieve competitive performances compared with previous models trained on all in-domain dataset, but also significantly outperform models trained on *scr10* by a large margin. The proposed models further show ability to adapt to a new domain using a limited amount of target domain data.

9.2.1. Ablation studies

The ablation studies (Table 9, *sec*.1, 2, 4, 5) demonstrate the contribution of two Critics, in which the models were assessed with either no Critics (*sec*.1) or both (*sec*.2) or only one (+ DC only in *sec*.4 and + SC only in *sec*.5). It is clearly shown that, in comparison to models trained without Critics in Table 9 *sec*.1, combining both Critics (*sec*.2) makes a substantial contribution to increasing the BLEU score and decreasing the slot error rate ERR by a large margin in every dataset pair. A comparison of model adapting from source Laptop domain between VIR-RALSTM without Critics (Laptop in *sec*.1) and VDANLG (Laptop in *sec*.2)

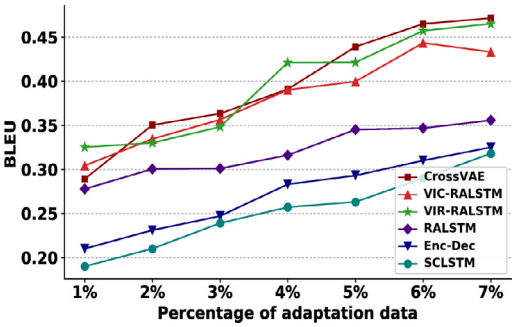


Fig. 6. Performance on **Laptop** domain with varied limited amount, from 1% to 7%, of the adaptation training data when adapting models pre-trained on [Restaurant+Hotel] union dataset.

Table 8
Results evaluated on **Target** domains by training models from scratch scenarios, *scr100* (in *sec*.1) and *scr10* (in *sec*.2).

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%
VIR-RALSTM (Ours)	0.8851	0.57%	0.7709	0.36%	0.5356	0.73%	0.5210	0.59%
VIC-RALSTM (Ours)	0.8811	0.49%	0.7651	0.06%	0.5350	0.88%	0.5192	0.56%
RALSTM Tran and Nguyen (2017a)	0.6855	22.53%	0.6003	17.65%	0.4009	22.37%	0.4475	24.47%
VIR-RALSTM (Ours)	0.7378	15.43%	0.6417	15.69%	0.4392	17.45%	0.4851	10.06%
VIC-RALSTM (Ours)	0.7998	8.67%	0.6838	6.86%	0.5040	5.31%	0.4932	3.56%

Table 9
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 models were assessed with either **no Critics** (*sec*.1) or both (**+ DC + SC** in *sec*.2) or only one (**+ DC only** in *sec*.4 and **+ SC only** in *sec*.5). **DC** and **SC** are stand for Domain Critic and Text Similarity Critic, respectively. **scr10** in *sec*.3: Training RALSTM and VIR-RALSTM models from *scratch* using only 10% of **Target** domain data. The results were averaged over 5 randomly initialized networks.

	Source\Target	Hotel		Restaurant		Tv		Laptop	
		BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	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%
	VIR-RALSTM	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%	–	–	–	–

evaluated on the target domain **Hotel** 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 slot error rate ERR, from 15.17% down to 2.89%. The trend is consistent across all the other domain pairs. These stipulate the necessity of the Critics and the adversarial domain adaptation algorithm in effective learning to adapt to a new domain, in which although both the RALSTM and VIR-RALSTM models perform well when providing sufficient in-domain training data (Table 8), the performances are extremely impaired when training from *scratch* with only limited amount of in-domain training data.

Table 9 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 (*sec*.4) are worse than those without Critics, or in some cases even worse than the VIR-RALSTM, 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 SC only (*sec*.5) helps the models achieve better results since it is aware of the sentence style when adapting to the target domain. These further demonstrate that the proposed variational-based models 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.

9.2.2. Adaptation *versus* scr100 *training scenario*

It is interesting to compare *adaptation* (Table 9, *sec.* 2) with *scr100* training scenario (Table 8). The VDANLG model shows its considerable ability to shift to another domain with a limited amount 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 (*sec*.2) 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 the slot error rate ERR results 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.

9.2.3. 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 the domain adaptation task becomes. This clearly

shows in Table 9, *sec.* 1, at **Hotel** column where the adaptation ability gets worse in terms of 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 9, *sec.* 1, 2). The distance of datasets is also shown via the differences of word-level distribution using word clouds in Fig. 1.

Table 9, *sec.* 1, 2 further demonstrate 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 9, *sec.* 2 shows that the adaptation VDANLG model trained on the source union dataset of Laptop and Tv ([L+T]) has better performance, at 82.52 BLEU and 2.87 ERR, than those models trained on the individual source dataset, such as Laptop (82.18 BLEU, 2.89 ERR) and Tv (82.51 BLEU, 4.89 ERR). Another example in Table 9, *sec.* 2 also shows that the adaptation VDANLG model trained on the source union dataset of Restaurant and Hotel ([R+H]) also has better results, at 51.97 BLEU and 2.58 ERR, than those models trained on the separate source dataset, such as Restaurant(50.43 BLEU, 2.99 ERR), and Hotel(51.64 BLEU, 3.25 ERR). The trend is mostly consistent across all other comparisons in different training scenarios. All these demonstrate that the proposed model can learn global semantics that can be efficiently transferred into new domains.

9.2.4. 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 10, despite the fact that the models were instead of 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 when the source domains (Hotel/Restaurant) 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 scores of slot error rate ERR (2.38/2.74).

9.2.5. Comparison on generated outputs

We present top responses generated for different scenarios from Laptop (Table 11, 12) and TV (Table 13) domains.

On the one hand, the VIR-RALSTM models (trained from *scratch* or trained adapting model from Source domains) produce outputs with a diverse range of error types, including missing, misplaced, redundant, wrong slots, or even spelling mistake information, leading to a very high score of the slot error rate ERR. Specifically, the VIR-RALSTM from *scratch* tends to make repeated slots and also many of the missing slots in generated outputs since the training data may be inadequate for the model to generally handle unseen dialog acts. Whereas the VIR-RALSTM models without Critics adapting trained from Source domains (denoted by $^\flat$ in Table 11, 12, 13) tend to generate the outputs with fewer error types than the model from *scratch* due to the VIR-RALSTM$^\flat$ 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 $^\sharp$) can effectively leverage the existing knowledge of source domains to better adapt to target domains. The VDANLG models can generate outputs in style of target domain with much fewer the error types compared with two above models. Furthermore, the VDANLG models seem to produce satisfactory utterances with more correct generated slots. For example, a sample outputted by the [R+H]$^\sharp$ in Table 11 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$^\sharp$ is a successful generation. Another samples in Table 12 generated by the Hotel$^\sharp$, Tv$^\sharp$, [R+H]$^\sharp$ models, and a sample generated by the Laptop$^\sharp$ in Table 13 are all fulfilled responses. An analysis of generated responses in Table 12 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 work acceptably well in the low-resource setting since they produce better results with a much lower score of the slot error rate ERR.

Table 10
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.

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%	–	–

Table 11
Comparison of top **Laptop** responses generated for different scenarios by adaptation training VIR-RALSTM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VIR-RALSTM from ***scratch***. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG♯ = VIR-RALSTM♭+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*')
Ref 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
VIR-RALSTM	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

9.3. Dual variational model for low-Resource in-Domain data

In this section, we again performed the dual variational model in different scenarios of low-resource setting, *i.e*, training models from scratch with 10% (*scr10*), 30% (*scr30*), and 100% (*scr100*) amount of in-domain data, and training domain adaptation models (*adaptation*). Overall, the proposed models obtained better performance regarding both the evaluation metrics across all domains in all training scenarios.

9.3.1. Ablation studies

The ablation studies (Table 14) demonstrate the contribution of each model components, in which we incrementally train the baseline RALSTM, the VIC-RALSTM (= RALSTM + Variational Inference), the DualVAE (= VIC-RALSTM + Variational CNN-DCNN), and

Table 12
Comparison of top **Laptop** responses generated for different scenarios by adaptation training VIR-RALSTM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VIR-RALSTM from ***scratch***. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG♯ = VIR-RALSTM♭+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*')
Ref 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*
VIR-RALSTM	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]

Table 13
Comparison of top **Tv** responses generated for different scenarios by adaptation training VIR-RALSTM (denoted by ♭) and VDANLG (denoted by ♯) models from Source domains, and by training VIR-RALSTM from ***scratch***. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. VDANLG♯ = VIR-RALSTM♭+SC+DC.

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

the CrossVAE (= DualVAE + Cross training) models. Generally, while all models can work well when there are sufficient training datasets, the performances of the proposed models also increase as increasing the proposed model components. The trend is consistent across all training cases no matter how much the training data was provided. Take, for example, the *scr100* scenario in which the CrossVAE model mostly outperformed all the previous baselines with regard to the BLEU and the slot error rate ERR scores.

On the other hand, the previous methods have extremely impaired performances regarding low BLEU score and high slot error rate ERR when training the models from *scratch* with insufficient in-domain data (*scr10*). In contrast, by integrating the variational inference, the VIC-RALSTM model can significantly improve the BLEU score from 68.55 to 79.98, and also reduce the slot error rate ERR by a large margin, from 22.53 to 8.67, compared to the baseline RALSTM model. Moreover, the proposed models have much better performance over the previous models in the *scr10* scenario since the CrossVAE, and the DualVAE models obtain the best and second best results, respectively. The CrossVAE model trained on *scr10* scenario, in some cases, achieved results which close to those of the HLSTM, SCLSTM, and Enc-Dec models trained on all in-domain data (*scr100*) scenario. Take, for example, the most challenge dataset *Laptop* and *Tv*, in which the DualVAE and CrossVAE obtained competitive results in terms of the BLEU score, at 50.16 and 50.85 respectively, which close to those of the HLSTM (51.30 BLEU), SCLSTM (51.09 BLEU), and Enc-Dec (51.01 BLEU), while the results regardless the slot error rate ERR scores are also close to those of the previous or even better in some cases, for example pairs of CrossVAE (2.86 ERR) and

Table 14
Results evaluated on four domains by training models from *scratch* with *10%*, *30%*, and *100%* in-domain data, respectively. The results were averaged over 5 randomly initialized networks. The **bold** and ***italic*** faces denote the best and second best models in each training scenario, respectively.

	Model	Hotel		Restaurant		Tv		Laptop	
		BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
scr100	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%**
	VIC-RALSTM (Ours)	0.8811	0.49%	0.7651	**0.06%**	0.5350	0.88%	0.5192	0.56%
	DualVAE (Ours)	0.8813	**0.33%**	0.7695	0.29%	0.5359	0.81%	0.5211	0.91%
	CrossVAE (Ours)	***0.8896***	0.72%	**0.7786**	0.54%	**0.5383**	**0.48%**	**0.5240**	***0.50%***
scr10	HLSTM Wen et al. (2015a)	0.7483	8.69%	0.6586	6.93%	0.4819	9.39%	0.4813	7.37%
	SCLSTM Wen et al. (2015b)	0.7626	17.42%	0.6446	16.93%	0.4290	31.87%	0.4729	15.89%
	Enc-Dec Wen et al. (2016b)	0.7370	23.19%	0.6174	23.63%	0.4570	21.28%	0.4604	29.86%
	RALSTM Tran and Nguyen (2017a)	0.6855	22.53%	0.6003	17.65%	0.4009	22.37%	0.4475	24.47%
	VIC-RALSTM (Ours)	0.7998	8.67%	0.6838	***6.86%***	0.5040	5.31%	0.4932	3.56%
	DualVAE (Ours)	***0.8022***	***6.61%***	***0.6926***	7.69%	***0.5110***	***3.90%***	***0.5016***	***2.44%***
	CrossVAE (Ours)	**0.8103**	**6.20%**	**0.6969**	**4.06%**	**0.5152**	**2.86%**	**0.5085**	**2.39%**
scr30	HLSTM Wen et al. (2015a)	0.8104	6.39%	0.7044	2.13%	0.5024	5.82%	0.4859	6.70%
	SCLSTM Wen et al. (2015b)	0.8271	6.23%	0.6825	4.80%	0.4934	7.97%	0.5001	3.52%
	Enc-Dec Wen et al. (2016b)	0.7865	9.38%	0.7102	13.47%	0.5014	9.19%	0.4907	10.72%
	RALSTM Tran and Nguyen (2017a)	0.8334	4.23%	0.7145	2.67%	0.5124	3.53%	0.5106	2.22%
	VIC-RALSTM (Ours)	***0.8553***	2.64%	0.7256	***0.96%***	0.5265	**0.66%**	***0.5117***	2.15%
	DualVAE (Ours)	0.8534	***1.54%***	***0.7301***	2.32%	***0.5288***	1.05%	0.5107	***0.93%***
	CrossVAE (Ours)	**0.8585**	**1.37%**	**0.7479**	**0.49%**	**0.5307**	***0.82%***	**0.5154**	**0.81%**

Enc-Dec (3.38 ERR), or DualVAE (2.44 ERR) and Enc-Dec (4.24 ERR). These indicate that the proposed model can efficiently encode useful information into the latent variable to better generalize to the unseen dialogue acts.

The *scr30* section further confirms the effectiveness of the proposed methods, in which the CrossVAE and DualVAE still mostly rank the best and second-best models compared with the baselines. The proposed models also show superior ability in leveraging the existing small training data to obtain very good performances, which are in many cases even better than those of the previous methods trained on 100% of in-domain data. Take **Tv** domain, for example, in which the CrossVAE in *scr30* achieves a good result in terms of BLEU and slot error rate ERR score, at 53.07 BLEU and 0.82 ERR, that are not only competitive to the RALSTM (53.76 BLEU, 0.65 ERR), but also outperform the previous models in *scr100* training scenario, such as HLSTM (52.40 BLEU, 2.65 ERR), SCLSTM (52.35 BLEU, 2.41 ERR), and Enc-Dec (51.42 BLEU, 3.38 ERR). This further indicates the need of the integrating with variational inference, the additional auxiliary auto-encoding, as well as the joint and cross training.

9.3.2. Model comparison on unseen domain

In this experiment, we trained four models (Enc-Dec, SCLSTM, RALSTM and CrossVAE) from *scratch* in the most difficult unseen Laptop domain with an increasingly varied proportion of training data, start from 10% to 100%. The results are shown in Fig. 7. It clearly sees that the BLEU score increases and the slot error ERR decreases as the models are trained on more data. The CrossVAE model is clearly better than the previous models (Enc-Dec, SCLSTM, RALSTM) in all cases. While the performance of the CrossVAE, RALSTM model starts to saturate around 30% and 50%, respectively, the Enc-Dec model seems to continue getting better as providing more training data. The figure also confirms that the CrossVAE trained on 30% of data can achieve a better performance compared to those of the previous models trained on 100% of in-domain data.

9.3.3. Domain adaptation

We further examine the domain scalability of the proposed methods by training the CrossVAE and SCLSTM models on adaptation scenarios, in which we first trained the models on out-of-domain data, and then fine-tuned the model parameters by using a small amount (10%) of in-domain data. The results are shown in Table 15.

In terms of distance of dataset pairs as described in Section 9.2.3, both SCLSTM (*sec.* 1) and CrossVAE (*sec.* 2, 3) models can take advantage of "*close*" dataset pairs, *i.e.*, Restaurant ↔ Hotel, and Tv ↔ Laptop, to achieve better performances compared to those of

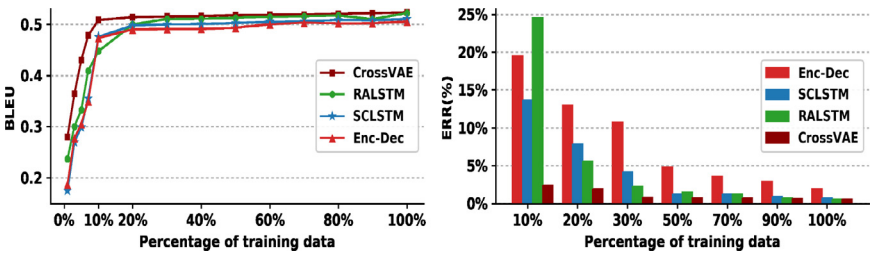


Fig. 7. Performance comparison of the models trained on Laptop domain.

Table 15
Results evaluated on **Target** domains: by *adaptation* training SCLSTM model from 100% (denoted as ♭) of Source data, and the CrossVAE model from 30% (denoted as ♯), 100% (denoted as ξ) of Source data. The scenario used only 10% amount of the **Target** domain data. The last two row show results by training the CrossVAE model on the *scr10* and semi-supervised learning, respectively.

Source\ Target	Hotel		Restaurant		Tv		Laptop	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
Hotel$^{\flat}$	–	–	0.6243	11.20%	0.4325	29.12%	0.4603	22.52%
Restaurant$^{\flat}$	0.7329	29.97%	–	–	0.4520	24.34%	0.4619	21.40%
Tv$^{\flat}$	0.7030	25.63%	0.6117	12.78%	–	–	0.4794	11.80%
Laptop$^{\flat}$	0.6764	39.21%	0.5940	28.93%	0.4750	14.17%	–	–
Hotel$^{\sharp}$	–	–	0.7138	2.91%	0.5012	5.83%	0.4949	1.97%
Restaurant$^{\sharp}$	0.7984	4.04%	–	–	0.5120	3.26%	0.4947	1.87%
Tv$^{\sharp}$	0.7614	5.82%	0.6900	5.93%	–	–	0.4937	1.91%
Laptop$^{\sharp}$	0.7804	5.87%	0.6565	6.97%	0.5037	3.66%	–	–
Hotel$^{\xi}$	–	–	0.6926	3.56%	0.4866	11.99%	0.5017	3.56%
Restaurant$^{\xi}$	0.7802	3.20%	–	–	0.4953	3.10%	0.4902	4.05%
Tv$^{\xi}$	0.7603	8.69%	0.6830	5.73%	–	–	0.5055	2.86%
Laptop$^{\xi}$	0.7807	8.20%	0.6749	5.84%	0.4988	5.53%	–	–
CrossVAE (*scr10*)	0.8103	6.20%	0.6969	4.06%	0.5152	2.86%	0.5085	2.39%
CrossVAE (*semi-U50-L10*)	0.8144	6.12%	0.6946	3.94%	0.5158	2.95%	0.5086	1.31%

the "*different*" dataset pairs, *i.e.* Latop ↔ Restaurant. This can be seen in Table 15, *sec.* 1, at Restaurant column where the adaptation ability gets worse in terms of decreasing the BLEU score, at 62.43, 61.17, and 59.40, respectively, and increasing the ERR score, at 11.20%, 12.78%, and 28.93%, respectively, alongside the order of Hotel → Tv → Laptop datasets. The SCLSTM (denoted by ♭) is limited to scale to a new domain in terms of having very low BLEU and high ERR scores. This adaptation scenario along with the *scr10* and *scr30* demonstrate that the SCLSTM can not work when having a low-resource setting of in-domain training data.

On the other hand, the CrossVAE model again show ability in leveraging the out-of-domain data to better adapt to a new domain. Especially in the case where Laptop, which is a most difficult unseen domain, is the target domain the CrossVAE model can obtain good results irrespective of low slot error rate ERR, around 1.90%, and high BLEU score, around 50.00 points. Surprisingly, the CrossVAE model trained on *scr10* scenario in some cases achieves better performance compared to those in adaptation scenario first trained with 30% out-of-domain data (denoted by ♯) which is also better than the adaptation model trained on 100% out-of-domain data (denoted by ♮).

A preliminary experiments on semi-supervised training is also conducted, in which we trained the CrossVAE model with the same 10% in-domain *labeled* data as in the other scenarios and 50% in-domain *unlabeled* data by keeping only the utterances **u** in a given input pair of dialogua act-utterance (**d, u**) (denoted by *semi-U50-L10*). The results showed CrossVAE's ability in leveraging the unlabeled data to achieve better results compared to those in *scratch* and *adaptation* scenarios. All these stipulate that the proposed models can perform acceptably well in both cases from scratch and domain adaptation where the in-domain training data is in short supply.

9.3.4. Comparison on generated outputs

We present top responses generated for different scenarios from TV (Table 16, 17) and Laptop (Table 18) domains, which further demonstrate the effectiveness of the proposed methods.

On the one hand, previous models trained on *scr10, scr30* scenarios produce a diverse range of the outputs' error types, including missing, misplaced, redundant, wrong slots, or spelling mistake information, resulting in a very high score of the slot error rate ERR. The Enc-Dec, HLSTM and SCLSTM models, for example, in Table 16 and 17 tend to generate outputs with redundant slots (*i.e. SLOT_HDMIPORT, SLOT_NAME*, SLOT_FAMILY), missing slots (*i.e.* [l7 family], [4 hdmi port -s]), or even in some cases produce some irrelevant slots (*i.e. SLOT_AUDIO, eco rating*), resulting in inadequate utterances.

On the other hand, the proposed models can effectively leverage the knowledge from only few of the existing training instances to better generalize to the unseen dialogue acts, leading to satisfactory responses. For example in Table 16, the proposed methods can generate adequate number of the required slots, resulting in fulfilled utterances (DualVAE-10, CrossVAE-10, DualVAE-30, CrossVAE-30), or acceptable outputs with much fewer error information, *i.e.* mis-ordered slots in the generated utterances (VIC-RALSTM-30).

For a much easier dialogue act in Table 17, previous models still produce some error outputs, whereas the proposed methods seem to form some specific slots into phrase in concise outputs. For example, instead of generating "the *proteus 73* is a *television*" phrase, the proposed models tend to concisely produce "the *proteus 73 television*".

The trend is mostly consistent in Table 18.

Table 16
Comparison of top **Tv** responses generated for different models in different scenarios. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. Model-X where X is amount of training data, *i.e.* 10%, 30%, or 100%.

Model	Generated Responses from TV Domain
DA 1	compare(name='*typhon 45*'; hdmiport='*2*'; family='*l2*'; name='*hades 48*'; hdmiport='*4*'; family='*l7*')
Reference 1	Compared to *typhon 45* which has *2 hdmi port -s* and is in the *L2 product family, hades 48* has *4 hdmi port -s* and is in the *L7 product family*. Which one do you prefer ?
Enc-Dec-10	the *typhon 45* is in the *l2 product family* and has *2 hdmi port -s* and is in the *l7 product family* with *4 hdmi port -s*, the *hades 48* is in the SLOT_FAMILY product family with a *SLOT_AUDIO*
HLSTM-10	the *typhon 45* is a great *eco rating*, the *hades 48* is in the *l2 family* with *2 hdmi port -s* [l7 family] [4 hdmi port -s]
SCLSTM-10	the *typhon 45* is the *hades 48* with *2 hdmi port* in the *l2 family*, the *SLOT_NAME* has *4 hdmi port -s* and *SLOT_HDMIPORT* hdmi port [l7 family]
VIC-RALSTM-10	the *typhon 45* has *2 hdmi port -s* and the *hades 48* is in the *l2 family* and has *4 hdmi port -s* [l7 family]
DualVAE-10	the *typhon 45* has *2 hdmi port -s* and is in the *l2 family* while the *hades 48* has *4 hdmi port -s* and is in the *l7 family* [OK]
CrossVAE-10	the *typhon 45* is in the *l2 family* with *2 hdmi port -s* while the *hades 48* has *4 hdmi port -s* and is in the *l7 family* [OK]
Enc-Dec-30	the *typhon 45* has *2 hdmi port -s*, the *hades 48* has *4 hdmi port -s*, the *SLOT_NAME* has *SLOT_HDMIPORT* hdmi port [l2 family] [l7 family]
HLSTM-30	the *typhon 45* is in the *l2 product family* with *2 hdmi port -s*, whereas the *hades 48* has *4 hdmi port* [l7 family]
SCLSTM-30	the *typhon 45* has *2 hdmi port -s*, the *hades 48* is in the *l2 product family*. [l7 family] [4 hdmi port -s]
VIC-RALSTM-30	the *typhon 45* has *2 hdmi port -s*, the *hades 48* is in the *l2 product family* and has *4 hdmi port -s* in *l7 family*
DualVAE-30	which do you prefer, the *typhon 45* in the *l2 product family* with *2 hdmi port -s*. the *hades 48* is in the *l7 family* with *4 hdmi port -s*. [OK]
CrossVAE-30	the *typhon 45* has *2 hdmi port -s* and in the *l2 family* while the *hades 48* has *4 hdmi port -s* and is in the *l7 family*. which item do you prefer [OK]
CrossVAE-100	the *typhon 45* has *2 hdmi port -s* and is in the *l2 product family*. the *hades 48* has *4 hdmi port -s* and is in the *l7 family* [OK]

Table 17
Comparison of top **Tv** responses generated for different models in different scenarios. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. Model-X where X is amount of training data, *i.e.* 10%, 30%, or 100%.

Model	Generated Responses from TV Domain
DA 2	recommend(name='*proteus 73*'; type='*television*'; price='*1500 dollars*'; audio='*nicam stereo*'; hdmiport='*2*')
Reference 2	*proteus 73* is a nice *television*. its price is *1500 dollars*, its audio is *nicam stereo*, and it has *2 hdmi port -s*.
Enc-Dec-10	the *proteus 73* is a great *television* with a *nicam stereo* and *2 hdmi port -s* [1500 dollars]
HLSTM-10	the *proteus 73* is a *television* with *2 hdmi port -s* and comes with a *nicam stereo* and *costs 1500 dollars* [OK]
SCLSTM-10	the *proteus 73* is a nice *television* with *nicam stereo* and *2 hdmi port -s* [1500 dollars]
VIC-RALSTM-10	the *proteus 73 television* has a *nicam stereo* and *2 hdmi port -s* and *costs 1500 dollars* [OK]
DualVAE-10	the *proteus 73 television* has a *nicam stereo* and *2 hdmi port -s* and *costs 1500 dollars* [OK]
CrossVAE-10	the *proteus 73 television* has *2 hdmi port -s* and a *nicam stereo* and *costs 1500 dollars* [OK]
Enc-Dec-30	the *proteus 73 television* has *2 hdmi port -s* and *nicam stereo audio* for *1500 dollars* [OK]
HLSTM-30	the *proteus 73 television* has a *nicam stereo* and *2 hdmi port -s* and is priced at *1500 dollars* [OK]
SCLSTM-30	the *proteus 73* is a nice *television* with *nicam stereo* and *2 hdmi port -s*. it is priced at *1500 dollars* [OK]
VIC-RALSTM-30	the *proteus 73 television* has *2 hdmi port -s, nicam stereo audio*, and *costs 1500 dollars* [OK]
DualVAE-30	the *proteus 73 television* has *2 hdmi port -s* and *nicam stereo audio* and *costs 1500 dollars* [OK]
CrossVAE-30	the *proteus 73 television* has *2 hdmi port -s* and *nicam stereo audio* and *costs 1500 dollars* [OK]
CrossVAE-100	the *proteus 73 television* has *2 hdmi port -s, nicam stereo audio*, and *costs 1500 dollars* [OK]

10. Conclusion and future work

We have presented a Variational-based NLG (VNLG) framework tackling the NLG issues of having a low-resource setting data. Based on this framework, we first propose a novel adversarial VNLG which consists of two critics which are Domain and Text similarity critics in an adversarial training procedure, solving the first issue in domain adaptation. For the second issue of having limited in-domain data, we propose a dual variational model which is a combination of a variational-based generator and a variational CNN-DCNN. We extensively conducted the experiments of both proposed models in various training scenarios, such as domain adaptation and training models from scratch, with varied proportion of training data, across four different domains. The experimental results show that, while the former generator has the ability to perform acceptably well in a new, unseen domain using a limited amount of target domain data, the latter model shows its ability to work well when the training in-domain data is scarce. The proposed models further show a positive sign in unsupervised domain adaptation as well as in semi-supervised training manners, which would be a worthwhile study in the future.

Table 18
Comparison of top **Laptop** responses generated for different models in different scenarios. Errors are marked in colors ([missing], misplaced, redundant, wrong, spelling mistake information). [OK] denotes successful generation. Model-X where X is amount of training data, *i.e.* 10%, 30%, or 100%.

Model	Generated Responses from Laptop Domain
DA	compare(name='*satellite pallas 21*'; battery='*4 h*'; drive='*500 gb*'; name='*satellite dinlas 18*'; battery='*3.5 h*'; drive='*1 tb*')
Reference	compared to satellite *pallas 21* which can last *4 h* and has a *500 gb drive, satellite dinlas 18* can last *3.5 h* and has a *1 tb drive*. which one do you prefer
Enc-Dec-10	the *satellite pallas 21* has a *500 gb drive*, the *satellite dinlas 18* has a *4 h battery* life and a *3.5 h battery* life and a *SLOT_BATTERY* battery life and a *1 tb drive*
HLSTM-10	the *satellite pallas 21* has a *4 h battery* life and a *500 gb drive*. which one do you prefer [satellite pallas 18] [3.5 h battery] [1 tb drive]
SCLSTM-10	the *satellite pallas 21* has a *4 h battery*, and has a *3.5 h battery* life and a *500 gb drive* and a *1 tb drive* [satellite dinlas 18]
VIC-RALSTM-10	the *satellite pallas 21* has a *500 gb drive* and a *4 h battery* life. the *satellite dinlas 18* has a *3.5 h battery* life and a *SLOT_BATTERY* battery life [1 tb drive]
DualVAE-10	the *satellite pallas 21* has a *4 h battery* life and a *500 gb drive* and the *satellite dinlas 18* with a *3.5 h battery* life and *is good for business computing*. which one do you prefer [1 tb drive]
CrossVAE-10	the *satellite pallas 21* with *500 gb* and a *1 tb drive*. the *satellite dinlas 18* with a *4 h battery* and a *SLOT_DRIVE* drive. which one do you prefer [3.5 h battery]
Enc-Dec-30	the *satellite pallas 21* has a *500 gb drive* with a *1 tb drive* and is the *satellite dinlas 18* with a *SLOT_DRIVE* drive for *4 h -s*. which one do you prefer [3.5 h battery]
HLSTM-30	the *satellite pallas 21* is a *500 gb drive* with a *4 h battery* life. the *satellite dinlas 18* has a *3.5 h battery* life. which one do you prefer [1 tb drive]
SCLSTM-30	the *satellite pallas 21* has a *500 gb drive*. the *satellite dinlas 18* has a *4 h battery* life. the *SLOT_NAME* has a *3.5 h battery* life. which one do you prefer [1 tb drive]
VIC-RALSTM-30	which one do you prefer the *satellite pallas 21* with a *4 h battery* life, the *satellite dinlas 18* has a *500 gb drive* and a *3.5 h battery* life and a 1 tb drive. which one do you prefer
DualVAE-30	*satellite pallas 21* has a *500 gb drive* and a *4 h battery* life while the *satellite dinlas 18* with a *3.5 h battery* life and a *1 tb drive*. [OK]
CrossVAE-30	the *satellite pallas 21* has a *500 gb drive* with a *4 h battery* life. the *satellite dinlas 18* has a *1 tb drive* and a *3.5 h battery* life. which one do you prefer [OK]
CrossVAE-100	the *satellite pallas 21* has a *500 gb drive* with a *4 h battery* life, while the *satellite dinlas 18* has a *1 tb drive* and a *3.5 h battery* life. which one do you prefer [OK]

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This paper is based upon work supported in part by the Asian Office of Aerospace R&D (AOARD), Air Force Office of Scientific Research (Grant no. FA2386-19-1-4041). We would also like to thank the Associate Editor and the anonymous reviewers for their careful reading of our manuscript and thoughtful comments, which have helped us to improve our manuscript significantly.

References

Bengio, S., Vinyals, O., Jaitly, N., Shazeer, N., 2015. Scheduled sampling for sequence prediction with recurrent neural networks. Advances in Neural Information Processing Systems, pp. 1171–1179.
Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Jozefowicz, R., Bengio, S., 2015a. Generating sentences from a continuous space. arXiv:1511.06349.
Bowman, S.R., Vilnis, L., Vinyals, O., Dai, A.M., Józefowicz, R., Bengio, S., 2015. Generating sentences from a continuous space. CoRR. arXiv:1511.06349.
Chen, X., Tan, T., Liu, X., Lanchantin, P., Wan, M., Gales, M.J., Woodland, P.C., 2015. Recurrent neural network language model adaptation for multi-genre broadcast speech recognition. Sixteenth Annual Conference of the International Speech Communication Association.
Chung, J., Ahn, S., Bengio, Y., 2016. Hierarchical multiscale recurrent neural networks. arXiv:1609.01704.
Chung, J., Kastner, K., Dinh, L., Goel, K., Courville, A.C., Bengio, Y., 2015. A recurrent latent variable model for sequential data. Advances in Neural Information Processing Systems, pp. 2980–2988.
Cuayáhuitl, H., Dethlefs, N., Hastie, H., Liu, X., 2014. Training a statistical surface realiser from automatic slot labelling. Spoken Language Technology Workshop (SLT), 2014 IEEE. IEEE, pp. 112–117.
Dethlefs, N., 2017. Domain transfer for deep natural language generation from abstract meaning representations. IEEE Comput. Intell. Mag. 12 (3), 18–28. https://doi.org/10.1109/MCI.2017.2708558.
Gangireddy, S.R., Swietojanski, P., Bell, P., Renals, S., 2016. Unsupervised Adaptation of Recurrent Neural Network Language Models. .
Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., Lempitsky, V., 2016. Domain-adversarial training of neural networks. J. Mach. Learn. Res. 17 (59), 1–35.
Gašić, M., Mrkšić, N., Rojas-Barahona, L.M., Su, P.-H., Ultes, S., Vandyke, D., Wen, T.-H., Young, S., 2017. Dialogue manager domain adaptation using gaussian process reinforcement learning. Comput. Speech Lang. 45, 552–569.
Huang, Q., Deng, L., Wu, D., Liu, C., He, X., 2018. Attentive tensor product learning for language generation and grammar parsing. arXiv:1802.07089.
Keizer, S., Rieser, V., 2018. Towards learning transferable conversational skills using multi-dimensional dialogue modelling. arXiv:1804.00146.
Kingma, D. P., Welling, M., 2013. Auto-encoding variational bayes. arXiv:1312.6114.
Konstas, I., Lapata, M., 2013. A global model for concept-to-text generation. J. Artif. Intell. Res. 48, 305–346.
Levin, E., Narayanan, S., Pieraccini, R., Biatov, K., Bocchieri, E., Fabbrizio, G.D., Eckert, W., Lee, S., Pokrovsky, A., Rahim, M., et al., 2000. The at&t-darpa communicator mixed-initiative spoken dialog system. Sixth International Conference on Spoken Language Processing.
Mairesse, F., Gašić, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., Young, S., 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. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1552–1561.
Mairesse, F., Gašić, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., Young, S., 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. Association for Computational Linguistics, Stroudsburg, PA, USA, pp. 1552–1561.
Mairesse, F., Walker, M.A., 2011. Controlling user perceptions of linguistic style: trainable generation of personality traits. Comput. Linguist. 37 (3), 455–488.
Mathur, P., Ueffing, N., Leusch, G., 2018. Multi-lingual neural title generation for e-commerce browse pages. arXiv:1804.01041.
Miao, Y., Yu, L., Blunsom, P., 2016. Neural variational inference for text processing. International Conference on Machine Learning, pp. 1727–1736.
Mnih, A., Gregor, K., 2014. Neural variational inference and learning in belief networks. arXiv:1402.0030.
Mo, K., Zhang, Y., Yang, Q., Fung, P., 2017. Fine grained knowledge transfer for personalized task-oriented dialogue systems. arXiv:1711.04079.
Mrkšić, N., Séaghdha, D. O., Thomson, B., Gašić, M., Su, P.-H., Vandyke, D., Wen, T.-H., Young, S., 2015. Multi-domain dialog state tracking using recurrent neural networks. arXiv:1506.07190.
Neculoiu, P., Versteegh, M., Rotaru, M., Amsterdam, T.B., 2016. Learning text similarity with siamese recurrent networks. ACL 2016, p. 148.
Papineni, K., Roukos, S., Ward, T., Zhu, W.-J., 2002. Bleu: a method for automatic evaluation of machine translation. In: Proceedings of the 40th ACL. Association for Computational Linguistics, pp. 311–318.
Purushotham, S., Carvalho, W., Nilanon, T., Liu, Y., 2017. Variational adversarial deep domain adaptation for health care time series analysis.
Radford, A., Metz, L., Chintala, S., 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434.
Rezende, D. J., Mohamed, S., 2015. Variational inference with normalizing flows. arXiv:1505.05770.
Serban, I.V., Sordoni, A., Lowe, R., Charlin, L., Pineau, J., Courville, A.C., Bengio, Y., 2017. A hierarchical latent variable encoder-decoder model for generating dialogues.
Shi, Y., Larson, M., Jonker, C.M., 2015. Recurrent neural network language model adaptation with curriculum learning. Comput. Speech Lang. 33 (1), 136–154.
Sohn, K., Lee, H., Yan, X., 2015. Learning structured output representation using deep conditional generative models. Advances in Neural Information Processing Systems, pp. 3483–3491.
Tran, V.-K., Nguyen, L.-M., 2017. 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). Association for Computational Linguistics, Vancouver, Canada, pp. 442–451.
Tran, V.-K., Nguyen, L.-M., 2017b. Semantic refinement gru-based neural language generation for spoken dialogue systems. arXiv:1706.00134.
Tran, V.K., Nguyen, L.M., 2018. Adversarial domain adaptation for variational natural language generation in dialogue systems. COLING.Santa Fe, New-Mexico, USA
Tran, V.K., Nguyen, L.M., 2018. Dual latent variable model for low-resource natural language generation in dialogue systems. ConLL.Brussels, Belgium
Tran, V.-K., Nguyen, L.-M., Tojo, S., 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. Association for Computational Linguistics, Saarbracken, Germany, pp. 231–240.
Tseng, B.-H., Kreyssig, F., Budzianowski, P., Casanueva, I., Wu, Y.-C., Ultes, S., Gasic, M., 2018. Variational cross-domain natural language generation for spoken dialogue systems. In: Proceedings of the 19th Annual SIGdial Meeting on Discourse and Dialogue, pp. 338–343.
Tzeng, E., Hoffman, J., Saenko, K., Darrell, T., 2017. Adversarial discriminative domain adaptation. arXiv:1702.05464.
Walker, M.A., Rambow, O., Rogati, M., 2001. Spot: A trainable sentence planner. In: Proceedings of the 2nd NAACL. Association for Computational Linguistics, pp. 1–8.
Walker, M.A., Stent, A., Mairesse, F., Prasad, R., 2007. Individual and domain adaptation in sentence planning for dialogue. J. Artif. Intell. Res. 30, 413–456.

Wen, T.-H., Gašić, M., Kim, D., Mrkšić, N., Su, P.-H., Vandyke, D., Young, S., 2015. Stochastic Language Generation in Dialogue using Recurrent Neural Networks with Convolutional Sentence Reranking. In: Proceedings SIGDIAL. Association for Computational Linguistics.
Wen, T.-H., Gasic, M., Mrksic, N., Rojas-Barahona, L. M., Su, P.-H., Vandyke, D., Young, S., 2016a. Multi-domain neural network language generation for spoken dialogue systems. arXiv:1603.01232.
Wen, T.-H., Gašic, M., Mrkšic, N., Rojas-Barahona, L. M., Su, P.-H., Vandyke, D., Young, S., 2016b. Toward multi-domain language generation using recurrent neural networks.
Wen, T.-H., Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., Young, S., 2015. Semantically conditioned LSTM-based natural language generation for spoken dialogue systems. In: Proceedings of EMNLP. Association for Computational Linguistics.
Wen, T.-H., Miao, Y., Blunsom, P., Young, S., 2017. Latent intention dialogue models. arXiv:1705.10229.
Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., Yu, K., 2010. The hidden information state model: a practical framework for pomdp-based spoken dialogue management. Comput. Speech Lang. 24 (2), 150–174.
Young, S., Gašić, M., Thomson, B., Williams, J.D., 2013. Pomdp-based statistical spoken dialog systems: areview. Proc. IEEE 101 (5), 1160–1179.
Zhang, B., Xiong, D., Su, J., Duan, H., Zhang, M., 2016. Variational Neural Machine Translation. arXiv:1605.07869.
Zhang, Y., Shen, D., Wang, G., Gan, Z., Henao, R., Carin, L., 2017. Deconvolutional paragraph representation learning. Advances in Neural Information Processing Systems, pp. 4172–4182.
Zhao, H., Zhang, S., Wu, G., Costeira, J. P., Moura, J. M., Gordon, G. J., 2017. Multiple source domain adaptation with adversarial training of neural networks. arXiv:1705.09684.