Title	音声の話システムの自然言語生成のための深い学習に関する研究
Author(s)	Tran, Van Khanh
Citation	
Issue Date	2018-09
Type	Thesis or Dissertation
Text version	ETD
URL	http://hdl.handle.net/10119/15529
Rights	
Description	Supervisor:NGUYEN, Minh Le, 情報科学研究科, 博士

Doctoral Dissertation

A Study on Deep Learning for Natural Language Generation in Spoken Dialogue Systems

TRAN Van Khanh

Supervisor: Associate Professor NGUYEN Le Minh

School of Information Science
Japan Advanced Institute of Science and Technology

September, 2018

To my wife, my daughter, and my family.
Without whom I would never have completed this dissertation.

Abstract

Natural language generation (NLG) plays a critical role in spoken dialogue systems (SDSs) and aims at converting a meaning representation, *i.e.*, a dialogue act (DA), into natural language utterances. NLG process in SDSs can typically be split up into two stages: sentence planning and surface realization. Sentence planning decides the order and structure of sentence representation, followed by a surface realization that converts the sentence structure into appropriate utterances. Conventional methods to NLG rely heavily on extensive hand-crafted rules and templates that are time-consuming, expensive and do not generalize well. The resulting NLG systems, thus, tend to generate stiff responses, lacking several factors: adequacy, fluency and naturalness. Recent advances in data-driven and deep neural networks (DNNs) methods have facilitated investigation of NLG in the study. DNN methods to NLG for SDS have demonstrated to generate better responses than conventional methods concerning factors as mentioned above. Nevertheless, when dealing with the NLG problems, such DNN-based NLG models still suffer from some severe drawbacks, namely completeness, adaptability and low-resource setting data. Thus, the primary goal of this dissertation is to propose DNN-based generators to tackle the problems of the existing DNN-based NLG models.

Firstly, we present gating generators based on a recurrent neural network language model (RNNLM) to overcome the NLG problems of completeness. The proposed gates are intuitively similar to those in the Long short-term memory (LSTM) or Gated recurrent unit (GRU) to restrain the gradient vanishing and exploding. In our models, the proposed gates are in charge of sentence planning to decide "How to say it?", whereas the RNNLM forms a surface realization to generate surface texts. More specifically, we introduce three additional semantic cells based on the gating mechanism, into a traditional RNN cell. While a refinement cell is to filter the sequential inputs before RNN computations, an adjustment cell and an output cell are to select semantic elements and to gate a feature vector DA during generation, respectively. The proposed models further obtain state-of-the-art results over previous models regarding BLEU and slot error rate ERR scores.

Secondly, we propose a novel hybrid NLG framework to address the first two NLG problems, which is an extension of an RNN Encoder-Decoder incorporating with an attention mechanism. The idea of attention mechanism is to automatically learn alignments between features from source and target sentence during decoding. Our hybrid framework consists of three components: an encoder, an aligner, and a decoder, from which we propose two novel generators to leverage gating and attention mechanisms. In the first model, we introduce an additional cell into aligner cell by utilizing another attention or gating mechanisms to align and control the semantic elements produced by the encoder with a conventional attention mechanism over the input elements. In the second model, we develop a refinement adjustment LSTM (RALSTM) decoder to select, aggregate semantic elements and to form the required utterances. The hybrid generators not only tackle the NLG problems of completeness, achieving state-of-the-art performances over previous methods, but also deal with adaptability issue by showing an ability to

adapt faster to a new, unseen domain and to control feature vector DA effectively.

Thirdly, we propose a novel approach dealing with the problem of low-resource setting data in a domain adaptation scenario. The proposed models demonstrate an ability to perform acceptably well in a new, unseen domain by using only 10% amount of the target domain data. More precisely, we first present a variational generator by integrating a variational autoencoder into the hybrid generator. We then propose two critics, namely domain, and text similarity, in an adversarial training algorithm to train the variational generator via multiple adaptation steps. The ablation experiments demonstrated that while the variational generator contributes to learning the underlying semantic of DA-utterance pairs effectively, the critics play a crucial role in guiding the model to adapt to a new domain in the adversarial training procedure.

Fourthly, we propose another approach dealing with the problem of having low-resource in-domain training data. The proposed generators, which combines two variational autoencoders, can learn more efficiently when the training data is in short supply. In particularly, we present a combination of a variational generator with a variational CNN-DCNN, resulting in a generator which can perform acceptably well using only 10% to 30% amount of in-domain training data. More importantly, the proposed model demonstrates state-of-the-art performance regarding BLEU and ERR scores when training with all of the in-domain data. The ablation experiments further showed that while the variational generator makes a positive contribution to learning the global semantic information of pairs of DA-utterance, the variational CNN-DCNN play a critical role of encoding useful information into the latent variable.

Finally, all the proposed generators in this study can learn from unaligned data by jointly training both sentence planning and surface realization to generate natural language utterances. Experiments further demonstrate that the proposed models achieved significant improvements over previous generators concerning two evaluation metrics across four primary NLG domains and variants in a variety of training scenarios. Moreover, the variational-based generators showed a positive sign in unsupervised and semi-supervised learning, which would be a worthwhile study in the future.

Keywords: natural language generation, spoken dialogue system, domain adaptation, gating mechanism, attention mechanism, encoder-decoder, low-resource data, RNN, GRU, LSTM, CNN, Deconvolutional CNN, VAE.

Acknowledgements

I would like to thank my supervisor, Associate Professor Nguyen Le Minh, for his guidance and motivation. He gave me a lot of valuable and critical comments, advice and discussion, which foster me pursuing this research topic from the starting point. He always encourages and challenges me to submit our works to the top natural language processing conferences. During Ph.D. life, I learned many useful research experiences which benefit my future careers. Without his guidance and support, I would have never finished this research.

I would also like to thank the tutors in writing lab at JAIST: Terrillon Jean-Christophe, Bill Holden, Natt Ambassah and John Blake, who gave many useful comments on my manuscripts. I greatly appreciate useful comments from committee members: Professor Satoshi Tojo, Associate Professor Kiyoaki Shirai, Associate Professor Shogo Okada, and Associate Professor Tran The Truyen.

I must thank my colleagues in Nguyen's Laboratory for their valuable comments and discussion during the weekly seminar. I owe a debt of gratitude to all the members of the Vietnamese Football Club (VIJA) as well as the Vietnamese Tennis Club at JAIST, of which I was a member for almost three years. With the active clubs, I have the chance playing my favorite sports every week, which help me keep my physical health and recover my energy for pursuing research topic and surviving on the Ph.D. life.

I appreciate anonymous reviewers from the conferences who gave me valuable and useful comments on my submitted papers, from which I could revise and improve my works. I am grateful for the funding source that allowed me to pursue this research: The Vietnamese Government's Scholarship under the 911 Project "Training lecturers of Doctor's Degree for universities and colleges for the 2010-2020 period".

Finally, I am deeply thankful to my family for their love, sacrifices, and support. Without them, this dissertation would never have been written. First and foremost I would like to thank my Dad, Tran Van Minh, my Mom, Nguyen Thi Luu, my younger sister, Tran Thi Dieu Linh, and my parents in law for their constant love and support. This last word of acknowledgment I have saved for my dear wife Du Thi Ha and my lovely daughter Tran Thi Minh Khue, who always be on my side and encourage me to look forward to a better future.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Natural Language Generation (NLG) is the subfield of artificial intelligence and computational linguistics that is concerned with the construction of computer systems that can produce understandable texts in English or other human languages from some underlying non-linguistic representations (Reiter et al., 2000). The objective of NLG systems generally is to produce coherent natural language texts which satisfy a set of one or more communicative goals which describe the purpose of the text to be generated. NLG is also an essential component in a variety of *text-to-text* applications, including machine translation, text summarization, question answering; and *data-to-text* applications, including image captioning, weather and financial reporting, and spoken dialogue systems. This thesis mainly focuses on tackling NLG problems in spoken dialogue systems.

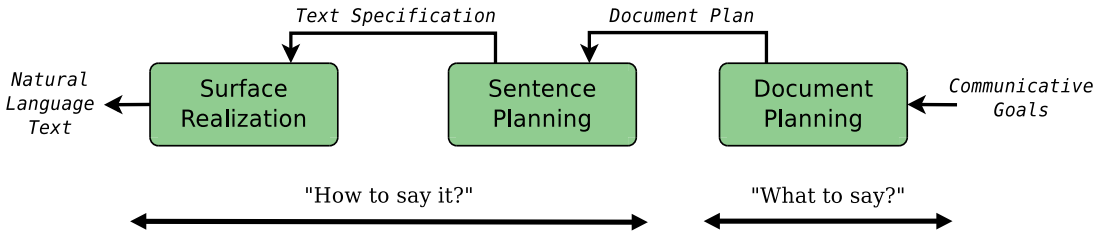


Figure 1.1: NLG system architecture.

Conventional NLG architecture consists of three stages (Reiter et al., 2000), namely *document planning*, *sentence planning*, and *surface realization*. Three stages are connected into a pipeline, in which the output of document planning is the input to sentence planning, and the output of sentence planning is the input to surface realization. While the sentence planning stage is to decide the "What to say?", the rest stages are in charge of deciding the "How to say it?". Figure 1.1 shows the traditional architecture of NLG systems.

- Document Planning (also called as Content Planning or Content Selection): This stage contains two concurrent subtasks. While the subtask *content determination* is to decide the "What to say?" information which should be communicated to the user, the *text planning* involves decision regarding the way this information should be rhetorically structured, such as the order and structuring.

- Sentence Planning (also called as Microplanning): This stage involves the process of deciding how the information will be divided into sentences or paragraphs, and how to make

them more fluent and readable by choosing which words, sentences, syntactic structures, and so forth will be used.

- Surface Realization: This stage involves the process of producing the individual sentences in a well-formed manner which should be a grammatical and fluent output.

A Spoken Dialogue System (SDS) is a complicated computer system which can converse with a human with voice. The spoken dialogue system in a pipeline architecture consists of a wide range of speech and language technologies, such as automatic speech recognition, natural language understanding, dialogue management, natural language generation, and text-to-speech synthesis. The pipeline architecture is shown in Figure 1.2.

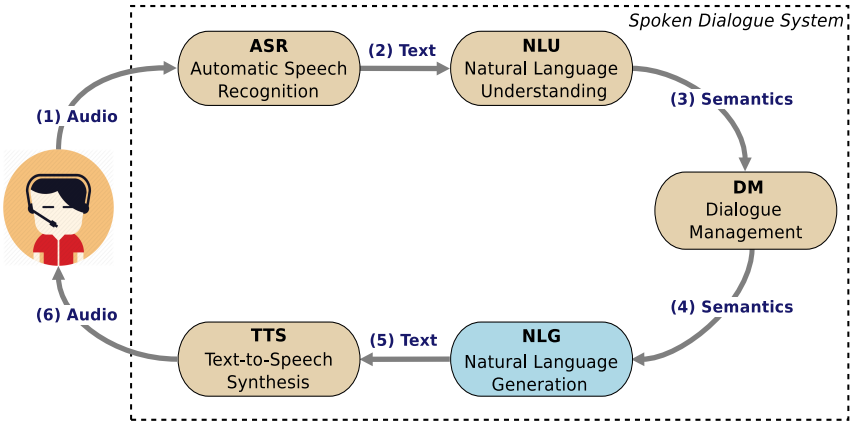


Figure 1.2: A pipeline architecture of a spoken dialogue system.

In the SDSs pipeline, the automatic speech recognizer (ASR) takes as input an acoustic speech signal (1) and decodes it into a string of words (2). The natural language understanding (NLU) component parses the speech recognition result and produces a semantic representation of the utterance (3). This representation is then passed to the dialogue manager (DM) whose task is to control the structure of the dialogue by handling the current dialogue state and making decisions about the system's behavior. This component generates a response (4) on a semantic representation of a communicative act from the system. The natural language generation (NLG) component takes as input a meaning representation from the dialogue manager and produces a surface representation of the utterance (5), which is then converted to the audio output (6) to the user by a text-to-speech synthesis (TTS) component. In the case of text-based SDSs, the speech recognition and speech synthesis can be left out.

Notwithstanding the architecture simplicity and modules reusability, there are several challenges in constructing NLG systems for SDSs. First, SDSs are typically developed for various specific domains (also called task-oriented SDS), *e.g.*, finding a hotel or a restaurant (Wen et al., 2015b), buying a laptop or a television (Wen et al., 2016a). Such systems often require large-scale corpora with a well-defined ontology which is necessarily a data structured representation that the dialogue system can converse. The process for collecting such large and specific domain datasets is extremely time-consuming and expensive. Second, NLG systems in the pipeline architecture easy suffer to a mismatch problem between "What" and "How" components (Meteer, 1991; Inui et al., 1992) since the early decisions may have unexpected effects downstream. Third, task-oriented SDSs typically use meaning representation (MR), *i.e.*,

dialogue acts (DAs[1]) (Young et al., 2010) to represent communicative actions of both user and system. NLG thus plays an essential role in SDSs since its task is to convert a given DA into natural language utterances. Last, NLG also has responsibility for adequate, fluent, and natural presentation of information provided by the dialogue system and has a profound impact on a user's impression of the system. Table 1.1 shows example pairs of DA-utterance in various NLG domains.

Table 1.1: 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*.

Traditional methods to NLG for SDSs still rely on extensive hand-tuning rules and templates, requiring expert knowledge of linguistic modeling, including rule-based methods (Duboue and McKeown, 2003; Danlos et al., 2011; Reiter et al., 2005), grammar-based methods (Reiter et al., 2000), corpus-based lexicalization (Bangalore and Rambow, 2000; Barzilay and Lee, 2002), template-based models (Busemann and Horacek, 1998; McRoy et al., 2001), or a trainable sentence planner (Walker et al., 2001; Ratnaparkhi, 2000; Stent et al., 2004). As a result, such NLG systems tend to generate stiff responses, lacking several factors: completeness, adaptability, adequacy, and fluency. Recently, taking advantages of advances in data-driven and deep neural network (DNN) approaches, NLG has received much attention in the study. DNN-based NLG systems have achieved better-generated results over traditional methods regarding completeness and naturalness as well as variability and scalability (Wen et al., 2015b, 2016b, 2015a). Deep learning based approaches have also shown promising performance in a wide range of applications, including natural language processing (Bahdanau et al., 2014; Luong et al., 2015a; Cho et al., 2014; Li and Jurafsky, 2016), dialogue systems (Vinyals and Le, 2015; Li et al., 2015), image processing (Xu et al., 2015; Vinyals et al., 2015; You et al., 2016; Yang et al., 2016), and so forth.

However, the aforementioned DNN-based methods suffer from some severe drawbacks when dealing with the NLG problems: (i) *completeness* that to ensure whether the generated utterances expresses the intended meaning in the dialogue act. Since DNN-based approaches for NLG are at the early stage, this issue leaves some rooms for improvement in terms of adequacy, fluency, and variability; (ii) *scalability/adaptability* that to examine whether the model can scale/adapt to a new, unseen domain since current DNN-based NLG systems also struggle to generalize well; and (iii) *low-resource* setting data that to examine whether the model can perform acceptably well when training on a modest amount of dataset. *Low-resource* training data can easily harm the performance of such NLG systems since the DNNs are often seen as data-hungry models. The primary goal of this thesis, thus, is to propose DNN-based architectures for solving NLG as mentioned above problems in SDSs.

[1]A dialogue act is a combination of an action type, *e.g.*, *request*, *recommend*, or *inform*, and a list of slot-value pairs extracted from corresponding utterance, *e.g.*, name='*Sushino*' and type='*restaurant*'.
A dialogue act example: *inform_count*(type='*hotel*'; count='*16*').

To achieve the goal, we pursue five primary objectives: (i) to investigate core DNN models, including recurrent neural networks (RNNs), convolutional neural networks (CNNs), encoder-decoder networks, variational autoencoder (VAE), word distributed representation, gating and attention mechanisms, and so forth, as well as the factors influencing the effectiveness of the DNN-based NLG models; (ii) to propose a DNN-based generator based on an RNN language model (RNNLM) and *gating* mechanism, that obtains better performance over previous NLG systems; (iii) to propose a DNN-based generator based on an RNN encoder-decoder, *gating* and *attention* mechanisms, which improves upon the existing NLG systems; (iv) to develop a DNN-based generator that performs acceptably well when training the generator from *domain adaptation* scenario on a *low-resource* of *target* data; (v) to develop a DNN-based generator that performs acceptably well when training the generator from *scratch* scenario on a *low-resource* of training data.

In this introductory chapter, we first present in Section 1.1 our motivation for the research. We then show our contributions in Section 1.2. Finally, we present thesis outline in Section 1.3.

1.1 Motivation for the research

This section discusses the two factors that motivate our research undertaken in this study. *First*, there is a need to enhance the current DNN-based NLG systems concerning naturalness, completeness, fluency, and variability, even though DNN methods have demonstrated impressive progress in improving the quality of SDSs. *Second*, there is a dearth of deep learning approaches for constructing open-domain NLG systems since such NLG systems have only been evaluated on specific domains. Such NLG systems cannot also scale to scale to a new domain and have poor performance when there is only a limited amount of training data. These are discussed in details in the following two Subsections, where Subsection 1.1.1 discusses the former motivating factor, and Subsection 1.1.2 discusses the latter motivation.

1.1.1 The knowledge gap

Conventional approaches to NLG follow a pipeline which typically breaks down the task into *sentence planning* and *surface realization*. Sentence planning is to map input semantic symbols onto a linguistic structure, *e.g.*, a tree-like or a template structure. Surface realization is then to convert the structure into an appropriate sentence. These approaches to NLG rely heavily on extensive hand-tuning rules and templates that are time-consuming, expensive and do not generalize well. The emergence of deep learning has recently impacted on the progress and success of NLG systems. Specifically, language model, which is based on RNNs and cast NLG as a sequential prediction problem, has illustrated ability to model long-term dependencies and to better generalize by using distributed vector representations for words.

Unfortunately, RNNs-based models in practice suffer from the *vanishing gradient* problem which is later overcome by LSTM and GRU networks by introducing sophisticated *gating mechanism*. The similar idea was applied to NLG resulting in a semantically conditioned LSTM-based generator (Wen et al., 2015b) that can learn a soft alignment between slot-value pairs and their realizations by bundling their parameters up via delexicalization procedure (see Section 2.3.2). Specifically, the gating generator can jointly learn semantic alignments and surface realization, in which the traditional LSTM/GRU cell is in charge of *surface realization*, while the *gating*-based cell acts as a *sentence planning*. Although the RNN-based NLG sys-

tems are easy to train and have better-generated outputs than previous methods, there are still rooms for improvement regarding adequacy, completeness, and fluency. This thesis addresses the need to enhance how better *gating* mechanism is integrated into RNN-based generators (see Chapter 3).

On the other hand, deep encoder-decoder networks (Vinyals and Le, 2015; Li et al., 2015), especially RNN encoder-decoder based models with *attention mechanism* have achieved significant performance in a variety of NLG related tasks, *e.g.*, neural machine translation (Bahdanau et al., 2014; Luong et al., 2015a; Cho et al., 2014; Li and Jurafsky, 2016), neural image captioning (Xu et al., 2015; Vinyals et al., 2015; You et al., 2016; Yang et al., 2016), and neural text summarization (Rush et al., 2015; Nallapati et al., 2016). *Attention-based* networks (Wen et al., 2016b; Mei et al., 2015) have also explored to tackle NLG problems with the ability to adapt faster to a new domain. The separate parameterization of slots and values under an *attention* mechanism provided encoder-decoder model (Wen et al., 2016b) signs to better *generalize* in the beginning. However, the influence of *attention* mechanism on NLG systems has remained unclear. The thesis investigates the need for improving *attention*-based NLG systems regarding the quality of generated outputs and ability to highly *scale* to multi-domains (see Chapter 4).

1.1.2 The potential benefits

Since the current DNN-based NLG systems have been only evaluated on specific domains, such as the laptop, restaurant or tv domains, constructing useful NLG models provides twofold benefits in *domain adaptation* training and *low-resource setting* training (see Chapter 5).

First, it enables the adaptation generator to achieve good performance on the target domain by leveraging knowledge from source data. 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 the target domain is assumed to have different characteristics from the source and have much smaller or even no labeled data. Hence, simply applying models trained on the source domain can lead to a worse performance in the target domain.

Second, it allows the generator to work acceptably well when there is a modest amount of in-domain data. The prior DNN-based NLG systems have proved to work well when providing a sufficient in-domain data, whereas a modest training data can harm the model performance. The latter poses a need of deploying a generator that can perform acceptably well on a *low-resource setting* dataset.

1.2 Contributions

Our main contributions of this thesis are summarized as follows:

- Proposing an effective *gating-based* RNN generator addressing the former knowledge gap. The proposed model empirically shows improved performance compared to previous methods;

- Proposing a novel *hybrid* NLG framework that combines *gating* and *attention* mechanisms, in which we introduce two *attention*- and *hybrid*-based generators addressing the latter knowledge gap. The proposed models achieve significant improvements over the previous methods across four domains;

- Proposing a *domain adaptation* generator which adapts faster to a new, unseen domain irrespective of scarce target resources, demonstrating the former potential benefit.
- Proposing a *low-resource setting* generator which performs acceptably well irrespective of a limited amount of in-domain resources, demonstrating the latter potential benefit.
- Illustrating the effectiveness of proposed generators by training on four different NLG domains and their variants in various scenarios, such as scratch, domain adaptation, semi-supervised training with different amount of data.

1.3 Thesis Outline

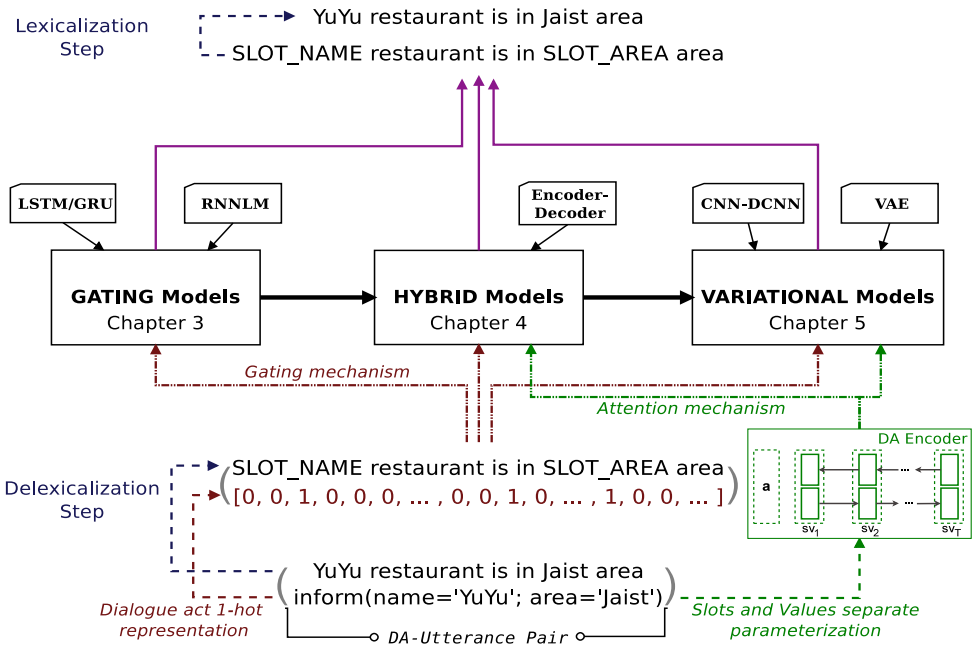


Figure 1.3: Thesis flow. Color arrows represent transformations going in and out of the generators in each chapter, while black arrow represents model hierarchy. Punch card with names, such as LSTM/GRU or VAE, represents core deep learning networks.

Figure 1.3 presents an overview of thesis chapters with an example, starting from the bottom with an input of Dialogue act-Utterance pair and ending at the top with an expected output after lexicalizing. While the utterance to be learned is delexicalized by replacing slot-value pair, *i.e.*, slot name '*area*' and slot value '*Jaist*', with a corresponding abstract token, *i.e.*, *SLOT_AREA*, the given dialogue act is represented by either using a *1-hot vector* (denoted by red dash arrow) or using a Bidirectional LSTM to separately parameterize its slots and values (denoted by green dash arrow and green box). The figure clearly shows that the gating mechanism is used in all proposed models in either a *solo* with proposed gating models in Chapter 3 or a *duet* with hybrid and variational models in Chapter 4 and 5, respectively. It is worth noting here that the decoder part of all proposed models in this thesis is mainly based on an RNN language model which is in charge of surface realization. On the other hand, while Chapter 3 presents an RNNLM generator which is based on gating mechanism and LSTM or GRU cells, Chapter 4 describes an RNN Encoder-Decoder in a mix of gating and attention mechanisms. Chapter 5 proposes

a variational generator which is a combination of the generator in Chapter 4 and a variety of deep learning models, such as convolutional neural networks (CNNs), deconvolutional CNNs and variational autoencoders.

Despite the strengths and potential benefits, the early DNN-based NLG architectures (Wen et al., 2015b, 2016b, 2015a) still have many shortcomings. In this thesis, we draw attention to three main problems pertaining to the existing DNN-based NLG models, namely *completeness*, *adaptability* and *low-resource* setting data. The thesis is organized as follows. Chapter 2 presents research background knowledge on NLG approaches by decomposing it into stages, whereas Chapters 3, 4, and 5 one by one address the three problems as mentioned earlier. The final Chapter 6 discusses main research findings and the future research direction for NLG. The content of Chapters 3, 4, 5 is briefly described as follows:

Gating Mechanism based NLG

This chapter presents a generator based on an RNNLM utilizing the *gating* mechanism to deal with the NLG problem of *completeness*.

Traditional approaches to NLG rely heavily on extensive hand-tuning templates and rules requiring linguistic modeling expertise, such as template-based (Busemann and Horacek, 1998; McRoy et al., 2001), grammar-based (Reiter et al., 2000), corpus-based (Bangalore and Rambow, 2000; Barzilay and Lee, 2002). Recent RNNLM-based approaches (Wen et al., 2015a,b) have shown promising results tackling the NLG problems of *completeness*, naturalness, and fluency. The methods cast NLG as a sequential prediction problem. To ensure the that generated utterances represent the intended meaning in a given DA, previous RNNLM-based models are further conditioned on a 1-hot DA vector representation. Such models leverage the strength of *gating mechanism* to alleviating the vanishing gradient problem in RNN-based models as well as keeping track of required slot-value pairs during generation. However, the models have trouble dealing with special slot-value pairs, such as *binary* slots and slots can take *dont_care* value. These slots cannot exactly match to words or phrase (see Hotel example in Table 1.1) in a delexicalized utterance (see Section 2.3.2). Following the line of research that models NLG problem in a unified architecture where the model can jointly train *sentence planning* and *surface realization*, in Chapter 3 we further investigate the effectiveness of *gating mechanism* and propose additional *gates* to address the *completeness* problem better. The proposed models not only demonstrate state-of-the-art performance over previous gating-based methods but also show signs to scale better to a new domain. *This chapter is based on the following papers (Tran and Nguyen, 2017b; Tran et al., 2017b; Tran and Nguyen, 2018d).*

Hybrid based NLG

This chapter proposes a novel generator on an attention RNN encoder-decoder (ARED) utilizing the *gating* and *attention* mechanisms to deal with the NLG problems of *completeness* and *adaptability*.

More recently, RNN Encoder-Decoder networks (Vinyals and Le, 2015; Li et al., 2015), especially the attentional based models (ARED) have not only been explored to solve the NLG issues (Wen et al., 2016b; Mei et al., 2015; Dušek and Jurčíček, 2016b,a) but have also shown improved performance on a variety of tasks, *e.g.*, image captioning (Xu et al., 2015; Yang et al., 2016), text summarization (Rush et al., 2015; Nallapati et al., 2016), neural machine translation (NMT) (Luong et al., 2015b; Wu et al., 2016). The attention mechanism (Bahdanau et al., 2014)

idea is to address sentence length problem in NLP applications, such as NMT, text summarization, text entailment by selectively focusing on parts of the source sentence or automatically learn alignments between features from source and target sentence during decoding. We further observe that while previous gating-based models (Wen et al., 2015a,b) are limited to generalize to the unseen domain (*scalability* issue), the current ARED-based generator (Wen et al., 2016b) has difficulty to prevent undesirable semantic repetitions during generation (*completeness* issue). Moreover, none of the existing models show significant advantage from out-of-domain data. To tackle these issues, in Chapter 4 we propose a novel ARED-based generation framework which is a *hybrid* model of gating and attention mechanisms. From this framework, we introduce two novel generators which are Encoder-Aggregator-Decoder (Tran et al., 2017a) and RALSTM (Tran and Nguyen, 2017a). Experiments showed that the *hybrid* generators not only achieve state-of-the-art performance compared to previous methods but also have an ability to *adapt* faster to a new domain and generate informative utterances. *This chapter is based on the following papers (Tran et al., 2017a; Tran and Nguyen, 2017a, 2018c).*

Variational Model for Low-Resource NLG

This chapter introduces novel generators based on *hybrid* generator integrating with a variational inference to deal with the NLG problems of *completeness* and *adaptability* and specifically *low-resource* setting data.

As mentioned, NLG systems for SDSs are typically developed for specific domains, such as reserving a flight, searching a restaurant, hotel, or buying a laptop, which requires a well-defined ontology dataset. The processes for collecting such well-defined annotated data are extremely time-consuming and expensive. Furthermore, the DNN-based NLG systems have obtained very good performance irrespective of providing adequate labeled datasets in the supervised learning manner, while *low-resource* setting data easily results in impaired performance models. In Chapter 5, we propose two approaches dealing with the problem of *low-resource* setting data. First, we propose an adversarial training procedure to train *variational* generator via multiple adaptation steps that enable the generator to learn more efficiently when the in-domain data is in short supply. Second, we propose a combination of two *variational* autoencoders that enables the *variational*-based generator to learn more efficiently in low-resource setting data. The proposed generators demonstrate state-of-the-art performance in both of rich and low-resource training data. *This chapter is based on the following papers (Tran and Nguyen, 2018a,b,e)*

Conclusion

In summary, this study has investigated various aspects in which the NLG systems have significantly improved performance. In this chapter, we provide main findings and discussions of this thesis. We believe that many NLG challenges and problems would be worth exploring in the future.

Chapter 2

Background

In this chapter, we present necessary background knowledge of the main topic in this dissertation, including NLG approaches, data processing, evaluation metrics, and so forth.

2.1 NLG Architecture for SDSs

This section briefly describes an NLG architecture for SDSs, which typically consists of three stages (Reiter et al., 2000), namely *document planning*, *sentence planning*, and *surface realization*. While the content determination phase decides "What to say" regarding domain concepts, the rest phases involve the decision of "How to say it" (see Chapter 1). However, in SDSs, *document planning* is handled by the dialogue manager (DM) which controls the current dialogue states and decides "What to say?" and "When to say it?" by yielding the meaning representation (MR). Whereas the NLG component only works with subtasks, *i.e.*, *sentence planning* and *surface realization*, in a two-step pipeline or joint approach to deciding "How to say it?" by mapping such MR into understandable texts. The MR, *i.e.*, dialogue act (Young et al., 2010), conveys the content to be expressed in the system's next dialogue turn. The NLG pipeline in SDSs is depicted in Figure 2.1.

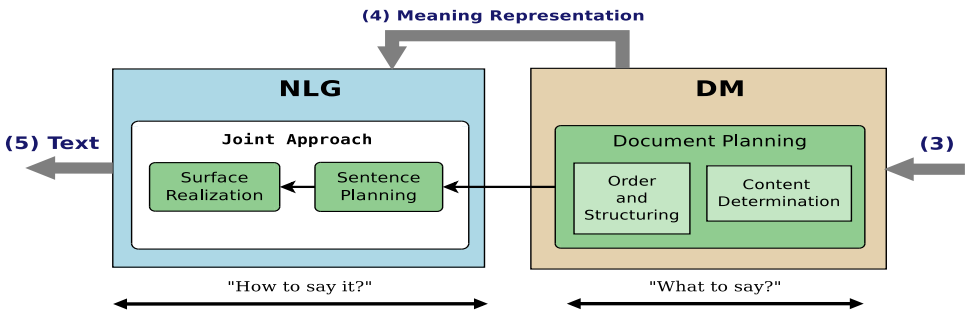


Figure 2.1: NLG pipeline in SDSs.

2.2 NLG Approaches

The following Subsections present most widely used NLG approaches in a broader view, ranging from traditional methods to recent approaches using neural networks.

2.2.1 Pipeline and Joint Approaches

While most NLG systems recently endeavor to learn generation from data, the choice between the pipeline and joint approach is often arbitrary and depends on specific domains and system architectures. A variety of systems follows the conventional pipeline tending to focus on subtasks, whether sentence planning (Stent et al., 2004; Paiva and Evans, 2005; Dušek and Jurcicek, 2015) or surface realization (Dethlefs et al., 2013) or both (Walker et al., 2001; Rieser et al., 2010), while others decide to follow a joint approach (Wong and Mooney, 2007; Konstas and Lapata, 2013). (Walker et al., 2004; Carenini and Moore, 2006; Demberg and Moore, 2006) followed pipeline to tailor user generation in the match multimodal dialogue system. (Oliver and White, 2004) proposed a model to present information in SDS by combining multi-attribute decision models, strategic document planning, dialogue management, and surface realization which incorporates prosodic features. Generators performing the joint approach employ various methods, e.g., factored language models (Mairesse and Young, 2014), inverted parsing (Wong and Mooney, 2007; Konstas and Lapata, 2013), or a pipeline of discriminative classifiers (Angeli et al., 2010). The pipeline approaches make the subtasks simpler, but feedbacks and revision in NLG system cannot be handled, whereas joint approaches do not require to explicitly model and handle intermediate structures (Konstas and Lapata, 2013).

2.2.2 Traditional Approaches

Traditionally, the most widely and common used NLG approaches are the *rule-based* (Duboue and McKeown, 2003; Danlos et al., 2011; Reiter et al., 2005; Siddharthan, 2010; Williams and Reiter, 2005) and grammar-based (Marsi, 2001; Reiter et al., 2000). In the document planning, (Duboue and McKeown, 2003) proposed three methods, such as exact matching, statistical selection, and rule induction to infer rules from indirect observations from the corpus, whereas in lexicalization, (Danlos et al., 2011) demonstrated a more practical rules-based approach which integrated into their EasyText NLG system, and (Siddharthan, 2010; Williams and Reiter, 2005) encompass the usage of choice rules. (Reiter et al., 2005) presented a model, which relies on consistent data-to-word rules, to convert a set of time phrases to linguistic equivalents through a fixed rule. However, these models required a comparison of the defined rules with expert suggested and corpus-derived phrases, whose processes are more resource expensive. It is also true that grammar-based methods for realization phase are so complex and learning to work with them takes a lot of time and effort (Reiter et al., 2000) because very large grammars need to be traversed for generation (Marsi, 2001).

Developing template-based NLG systems (McRoy et al., 2000; Busemann and Horacek, 1998; McRoy et al., 2001) is generally simpler than rule-based and grammar-based ones because the specification of templates requires less linguistic expertise than grammar rules. The template-based systems are also easier to adapt to a new domain since the templates are defined by hand, different templates can be specified for use on different domains. However, because of their use of handmade templates, they are most suitable for specific domains that are limited in size and subject to few changes. In addition, developing syntactic templates for a vast domain is very time-consuming and high maintenance costs.

2.2.3 Trainable Approaches

Trainable-based generation systems that have a trainable component tend to be easier to adapt to new domains and applications, such as trainable surface realization in NITROGEN (Langkilde

and Knight, 1998) and HALOGEN (Langkilde, 2000) systems, or trainable sentence planning (Walker et al., 2001; Belz, 2005; Walker et al., 2007; Ratnaparkhi, 2000; Stent et al., 2004). A trainable sentence planning proposed in (Walker et al., 2007) to adapt to many features of the dialogue domain and dialogue context, and to tailor to individual preferences of users. SPoT generator (Walker et al., 2001) proposed a trainable sentence planner via multiple steps with ranking rules. SPaRKy (Stent et al., 2004) used a tree-based sentence planning generator and then applied a trainable sentence planning ranker. (Belz, 2005) proposed a corpus-driven generator which reduces the need for manual corpus analysis and consultation with experts. This reduction makes it easier to build portable system components by combining the use of a base generator with a separate, automatically adaptable decision-making component. However, these trainable-based approaches still require a handmade generator to make decisions.

2.2.4 Corpus-based Approaches

Recently, NLG systems attempt to learn generation from data (Oh and Rudnicky, 2000; Barzilay and Lee, 2002; Mairesse and Young, 2014; Wen et al., 2015a). While (Oh and Rudnicky, 2000) trained n-gram language models for each DA to generate sentences and then selected the best ones using a rule-based re-ranker, (Barzilay and Lee, 2002) trained a corpus-based lexicalization on multi-parallel corpora which consisted of multiple verbalizations for related semantics. (Kondadadi et al., 2013) used an SVM re-ranker to further improve the performance of systems which extract a bank of templates from a text corpus. (Rambow et al., 2001) showed how to overcome the high cost of hand-crafting knowledge-based generation systems by employing statistical techniques. (Belz et al., 2010) developed a shared task in statistical realization based on common inputs and labeled corpora of paired inputs and outputs to reuse realization frameworks. The BAGEL system (Mairesse and Young, 2014), according to factored language models, treated the language generation task as a search for the most likely sequence of semantic concepts and realization phrases, resulting in a large variation found in human language using data-driven methods. The HALogen system (Langkilde-Geary, 2002) based on a statistical model, specifically an n-gram language model, that achieves both broad coverage and high-quality output as measured against an unseen section of the Penn Treebank. Corpus-based methods make the systems easier to build and extend to other domains. Moreover, learning from data enables the systems to imitate human responses more naturally, eliminates the needs of handcrafted rules and templates.

Recurrent Neural Networks (RNNs) based approaches have recently shown promising performance in tackling the NLG problems. For non-goal driven dialogue systems, (Vinyals and Le, 2015) proposed a sequence to sequence based conversational model that predicts the next sentence given the preceding ones. Subsequently, (Li et al., 2016a) presented a persona-based model to capture the characteristics of the speaker in a conversation. There have also been growing research interest in training neural conversation systems from large-scale of human-to-human datasets (Li et al., 2015; Serban et al., 2016; Chan et al., 2016; Li et al., 2016b). For task-oriented dialogue systems, RNN-based models have been applied for NLG as a joint training model (Wen et al., 2015a,b; Tran and Nguyen, 2017b) and an end-to-end training network (Wen et al., 2017a,b). (Wen et al., 2015a) combined a forward RNN generator, a CNN re-ranker, and a backward RNN re-ranker to generate utterances. (Wen et al., 2015b) proposed a semantically conditioned Long Short-term Memory generator (SCLSTM) which introduced a control sigmoid gate to the traditional LSTM cell to jointly learn the gating mechanism and language model. (Wen et al., 2016a) introduced an out-of-domain model which was trained

on counterfeited data by using semantically similar slots from the target domain instead of the slots belonging to the out-of-domain dataset. However, these methods require a sufficiently large dataset in order to achieve these results.

More recently, RNN Encoder-Decoder networks (Vinyals and Le, 2015; Li et al., 2015) and especially attentional RNN Encoder-Decoder (ARED)-based models have been explored to solve the NLG problems (Wen et al., 2016b; Mei et al., 2015; Dušek and Jurčíček, 2016b,a; Tran et al., 2017a; Tran and Nguyen, 2017a). (Wen et al., 2016b) proposed an attentive encoder-decoder based generator which computed the attention mechanism over the slot-value pairs. (Mei et al., 2015) proposed an ARED-based model by using two attention layers to train content selection and surface realization jointly.

Moving from a limited domain NLG to an open domain NLG raises some problems because of exponentially increasing semantic input elements. Therefore, it is important to build an open domain NLG that can leverage as much of abilities of knowledge from existing domains. There have been several works trying to solve this problem, such as (Mrkšić et al., 2015) utilizing the RNN-based model for multi-domain dialogue state tracking, (Williams, 2013; Gašić et al., 2015) adapting of SDS components to new domains. (Wen et al., 2016a) using a procedure to train multi-domain via multiple adaptation steps, in which a model was trained on counterfeited data by using semantically similar slots from the new domain instead of the slots belonging to the out-of-domain dataset, then fine tune the new domain on the out-of-domain trained model. While the RNN-based generators can prevent the undesirable semantic repetitions, the ARED-based generators show signs of better adapting to a new domain.

2.3 NLG Problem Decomposition

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

2.3.1 Input Meaning Representation and Datasets

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. The dataset ontology is shown in Table 2.1.

Table 2.1: 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***

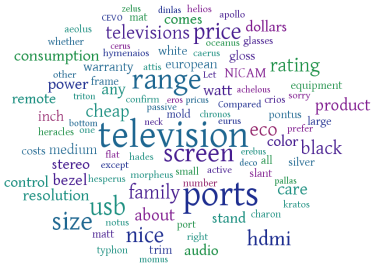
⋆ = overlap with **Restaurant** and **Hotel** domains, *italic* = slots can take *don't care* value, **bold** = binary slots.

In this study, we used 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.,

Table 2.2: Dataset statistics.

	Hotel	Restaurant	TV	Laptop
# train	3,223	3,114	4,221	7,944
# **10%** train	322	311	422	794
# **30%** train	966	933	1266	2382
# validation	1,075	1,039	1,407	2,649
# **10%** validation	107	103	140	264
# **30%** validation	321	309	420	792
# test	1,075	1,039	1,407	2,649
# distinct DAs	164	248	7,035	13,242
# act types	8	8	14	14
# slots	12	12	15	19

(a) Laptop domain.



(b) TV domain.

(c) Restaurant domain.

(d) Hotel domain.

Figure 2.2: Word clouds for testing set of the four original domains, in which font size indicates the frequency of words.

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 makes 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.

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

The dataset statistics is shown in Table 2.2. We also demonstrate the differences of word-level distribution using word clouds in Figure 2.2.

2.3.2 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 and slots that take *dont_care* cannot be delexicalized since their values cannot exactly match in the training corpus. Table 2.3 shows some examples of the delexicalization step.

Table 2.3: Delexicalization examples.

Hotel DA	inform_only_match(name = '*Red Victorian*' ; accepts_credit_cards = '*yes*' ; near = '*Haight*' ; has_internet = '*dont_care*')
Reference	The *Red Victorian* in the *Haight* area are the only hotel that *accepts credit cards* and *if the internet connection does not matter*.
Delexicalized Utterance	The *SLOT_NAME* in the *SLOT_AREA* area are the only hotel that *accepts credit cards* and *if the internet connection does not matter*.
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

2.3.3 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, 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 2.4 shows examples of the lexicalization process.

Table 2.4: Lexicalization examples.

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

2.3.4 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 Chapters 3, 4, 5 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., 2010; Konstas and Lapata, 2013) which requires domain specific constraints and explicit feature engineering. Examples in Tables 1.1, 2.3 and 2.4 show that correspondences between a DA and words or phrases in its output utterance are not always matched.

2.4 Evaluation Metrics

2.4.1 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 thesis used the cumulative *4*-gram BLEU score (also called BLEU-*4*) for the objective evaluation.

2.4.2 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:

$$\text{ERR} = (\mathbf{s}_m + \mathbf{s}_r)/\mathbf{N} \tag{2.1}$$

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 2.2). 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 2.5 demonstrates how to compute the ERR score with some examples. In this thesis, we adopted code from an NLG toolkit[1]to compute the two metrics BLEU and slot error rate ERR.

2.5 Neural based Approach

2.5.1 Training

This section describes the training procedure for proposed models in Chapters 3 and 4, in which the objective function was the negative log-likelihood and computed by:

$$\mathcal{L}(.) = -\sum_{t=1}^{T} \mathbf{y}_t^\top \log \mathbf{p}_t \tag{2.2}$$

where $\mathbf{y}_t$ is the ground truth token distribution, $\mathbf{p}_t$ is the predicted token distribution, T is length of the corresponding utterance.

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

Table 2.5: 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 2.2)
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 $= (1 + 1)/12 = 0.167$

Following the work of (Wen et al., 2015b), all proposed models were trained with a ratio of training, validation, and testing as 3:1:1. The models were initialized with a pre-trained Glove word embedding vectors (Pennington et al., 2014) and optimized by using stochastic gradient descent and back-propagation through time (Werbos, 1990). Early stopping mechanism was implemented to prevent over-fitting by using a validation set as suggested in (Mikolov, 2010). The proposed generators were trained by treating each sentence as a mini-batch with l_2 regularization added to the objective function for every 5 training examples. We performed 5 runs with different random initialization of the network, and the training is terminated by using early stopping. We then chose a model that yields the highest BLEU score on the validation set as reported in Chapter 3, 4. Since the trained models can differ depending on the initialization, we also report the results which were averaged over 5 randomly initialized networks.

2.5.2 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, conditioned on either representations of a given DA (Chapters 3 and 4), or both representations of a given DA and a latent variable z of variational-based generators (Chapter 5), 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:

$$\mathrm{R} = \mathcal{L}(.) + \lambda \mathrm{ERR} \tag{2.3}$$

where $\mathcal{L}(.)$ is cost of generator in the training phase, λ is a trade-off constant and is set to be large number to severely penalize nonsensical outputs. The slot error rate ERR (Wen et al., 2015b) is computed as in Eq. 2.1. We set λ to 100 to severely discourage the reranker from selecting utterances which contain either redundant or missing slots.

In the next chapter, we deploy our proposed *gating*-based generators which obtain state-of-the-art performances over previous *gating*-based models.

Chapter 3

Gating Mechanism based NLG

This chapter further investigates the *gating mechanism* in RNN-based models for constructing effective *gating-based* generators, tackling NLG issues of adequacy, completeness, and adaptability.

As previously mentioned, RNN-based approaches have recently improved performance in solving SDS language generation problems. Moreover, sequence to sequence models (Vinyals and Le, 2015; Li et al., 2015) and especially attention-based models (Bahdanau et al., 2014; Wen et al., 2016b; Mei et al., 2015) have been explored to solve the NLG problems. For task-oriented SDSs, RNN-based models have been applied for NLG in a joint training manner (Wen et al., 2015a,b) and an end-to-end training network (Wen et al., 2017b).

Despite the advantages and potential benefits, previous generators still suffer from some fundamental issues. The *first* issue of completeness and adequacy is that previous methods have lacked the ability to handle slots which cannot be directly delexicalized, such as binary slots (*i.e.*, *yes* and *no*) and slots that take *don't_care* value (Wen et al., 2015a), as well as to prevent the undesirable semantic repetitions (Wen et al., 2016b). The *second* issue of adaptability is that previous models have not generalized well to a new, unseen domain (Wen et al., 2015a,b). The *third* issue is that previous RNN-based generators often produce the next token based on information from the forward context, whereas the sentence may depend on backward context. As a result, such generators tend to generate nonsensical utterances.

To deal with the first issue that whether the generated utterance represents intended meaning of the given DA, previous RNN-based models were further conditioned on a 1-hot feature vector DA by introducing additional *gates* (Wen et al., 2015a,b). The *gating mechanism* has brought considerable benefits to not only mitigate the vanishing gradient problem in RNN-based models but also work as a *sentence planner* in the generator to keep track of the slot-value pairs during generation. However, there are still rooms for improvement with respect to all three issues.

Our objectives in this chapter are to investigate the *gating mechanism* to RNN-based generators. Our main contributions are summarized as follows:

- We present an effective way to construct gating-based RNN models, resulting in an end-to-end generator that empirically shows improved performance compared with previous gating-based approaches.

- We extensively conduct experiments to evaluate the models training from scratch on each in-domain dataset.

- We empirically assess the model ability to learn from multi-domain datasets by pooling

all existing training datasets, and then adapt to a new, unseen domain by feeding a limited amount of in-domain data.

The rest of this chapter is organized as follows. Sections 3.1.1, 3.1.2, 3.1.3 and 3.1.4 one by one present our gating-based generators addressing problems as mentioned earlier. We publish our work in (Tran and Nguyen, 2017b; Tran et al., 2017b) and (Tran and Nguyen, 2018d). Section 3.2 describes experimental setups while resulting analysis is presented in Section 3.3, in which the proposed methods significantly outperformed the previous gating- and attention-based methods regarding the BLEU and ERR scores. Experimental results also showed that the proposed generators could adapt faster to new domains by leveraging out-of-domain data. We give a summary and discussion in Section 3.4.

3.1 The Gating-based Neural Language Generation

The gating-based neural language generator proposed in this chapter is based on an RNN language model (Mikolov, 2010), which consists of three layers: an input layer, a hidden layer, and an output layer. The network takes input at each time step *t* as a 1-hot encoding $\mathbf{w}_t$ of a token[1] w_t which is conditioned on a recurrent hidden layer $\mathbf{h}_t$. The output layer $\mathbf{y}_t$ represents the probability distribution of the next token given previous token w_t and hidden $\mathbf{h}_t$. We can sample from this conditional distribution to obtain the next token in a generated string, and feed it as a next input to the generator. This process finishes when a stop sign is generated (Karpathy and Fei-Fei, 2015), or some constraints are reached (Zhang and Lapata, 2014). The network can produce a sequence of tokens which can be lexicalized[2] to form the required utterance. Moreover, to ensure that the generated utterance represents the intended meaning of the given DA, the generator is further conditioned on a vector $\mathbf{d}$, a 1-hot vector representation of DA. The following sections increasingly present in detail our methods by introducing five models: (i) a semantic Refinement GRU (RGRU) generator with two its variants, (ii) a Refinement-Adjustment-Output GRU (RAOGRU) generator with its ablation variant.

3.1.1 RGRU-Base Model

Inspired by work of (Wang et al., 2016) with an intuition: *Gating before computation*, we introduce a semantic gate before the RNN computation to refine the input tokens. With this intuition, instead of feeding an input token $\mathbf{w}_t$ to the RNN model at each time step t, the input token is filtered by a semantic gate which is computed as follows:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_{rd}\mathbf{d}) \\ \mathbf{x}_t &= \mathbf{r}_t \odot \mathbf{w}_t \end{aligned} \tag{3.1}$$

where $\mathbf{W}_{rd}$ is a trainable matrix to project the given DA representation into the word embedding space, $\mathbf{x}_t$ is new input. Here $\mathbf{W}_{rd}$ plays a role in sentence planning since it can directly capture which DA features are useful during the generation to encode the input information. The $\odot$ element-wise multiplication plays a part in word-level matching which not only learns the vectors similarity, but also preserves information about the two vectors. $\mathbf{r}_t$ is called a *refinement*

[1]Input texts are delexicalized in which slot values are replaced by its corresponding slot tokens.
[2]The process in which slot token is replaced by its value.

gate since the input tokens are refined by the DA information. As a result, we can represent the whole input sentence based on these refined inputs using RNN model.

In this study, we use GRU, which was recently proposed in (Bahdanau et al., 2014), as a building computational block for RNN, which is formulated as follows:

$$\begin{aligned}
\mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1}) \\
\mathbf{z}_t &= \sigma(\mathbf{W}_{zx}\mathbf{x}_t + \mathbf{W}_{zh}\mathbf{h}_{t-1}) \\
\tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{f}_t \odot \mathbf{W}_{hh}\mathbf{h}_{t-1}) \\
\mathbf{h}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t
\end{aligned} \tag{3.2}$$

where $\mathbf{W}_{fx}, \mathbf{W}_{fh}, \mathbf{W}_{zx}, \mathbf{W}_{zh}, \mathbf{W}_{hx}, \mathbf{W}_{hh}$ are weight matrices; $\mathbf{f}_t$, $\mathbf{z}_t$ are reset and update gate, respectively, and $\odot$ denotes for element-wise product. The semantic Refinement GRU (RGRU-Base) architecture is shown in Figure 3.1.

The output distribution of each token is defined by applying a softmax function g as follows:

$$P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{z}) = g(\mathbf{W}_{ho}\mathbf{h}_t) \tag{3.3}$$

where $\mathbf{W}_{ho}$ is learned linear projection matrix. At training time, we use the ground truth token for the previous time step in place of the predicted output. At test time, we implement a simple beam search to over-generate several candidate responses.

3.1.2 RGRU-Context Model

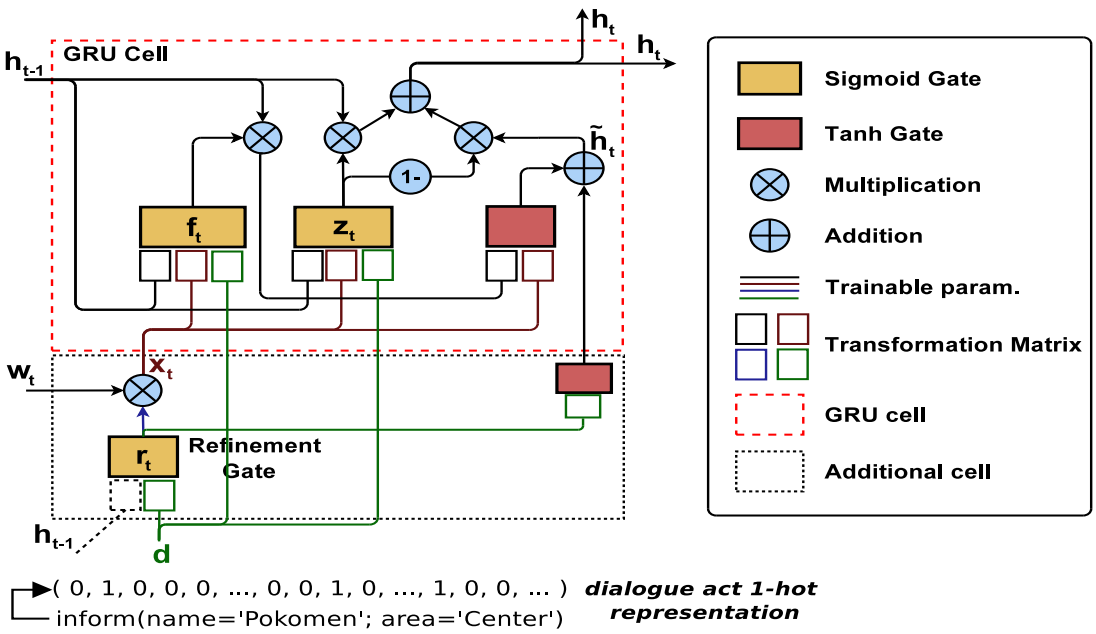


Figure 3.1: RGRU-Context cell. The red dashed box is a traditional GRU cell in charge of surface realization, while the black dotted box forms sentence planning based on a sigmoid control gate $\mathbf{r}_t$ and a 1-hot dialogue act $\mathbf{d}$. The contextual information $\mathbf{h}_{t-1}$ is imported into the refinement gate $\mathbf{r}_t$ via black dashed line and box. The RGRU-Base is achieved by omitting this link.

The RGRU-Base model uses only the DA information to gate the input sequence token by token. As a result, this gating mechanism may not capture the relationship between multiple

words. In order to import context information into the gating mechanism, Equation 3.1 is modified as follows:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_{rd}\mathbf{d} + \mathbf{W}_{rh}\mathbf{h}_{t-1}) \\ \mathbf{x}_t &= \mathbf{r}_t \odot \mathbf{w}_t \end{aligned} \tag{3.4}$$

where $\mathbf{W}_{rd}$ and $\mathbf{W}_{rh}$ are weight matrices. $\mathbf{W}_{rh}$ acts like a key phrase detector that learns to capture the pattern of generation tokens or the relationship between multiple tokens. In other words, the new input $\mathbf{x}_t$ consists of information of the original input token $\mathbf{w}_t$, the dialogue act $\mathbf{d}$, and the hidden context $\mathbf{h}_{t-1}$. $\mathbf{r}_t$ is called the *refinement* gate because the input tokens are refined by gating information from both the dialogue act $\mathbf{d}$ and the preceding hidden state $\mathbf{h}_{t-1}$. By taking advantage of gating mechanism from the LSTM model (Hochreiter and Schmidhuber, 1997) in which the gating mechanism is employed to solve the gradient vanishing and exploding problem, we propose to apply the dialog act representation $\mathbf{d}$ deeper into the GRU cell. Firstly, the GRU reset and update gates are computed under the influence of the dialogue act $\mathbf{d}$ and the refined input $\mathbf{x}_t$, and modified as follows:

$$\begin{aligned} \mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fd}\mathbf{d}) \\ \mathbf{z}_t &= \sigma(\mathbf{W}_{zx}\mathbf{x}_t + \mathbf{W}_{zh}\mathbf{h}_{t-1} + \mathbf{W}_{zd}\mathbf{d}) \end{aligned} \tag{3.5}$$

where $\mathbf{W}_{fd}$ and $\mathbf{W}_{zd}$ act as background detectors that learn to control the style of the generating sentence. Secondly, the candidate activation $\tilde{\mathbf{h}}_t$ is also modified to depend on the refinement gate:

$$\tilde{\mathbf{h}}_t = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{f}_t \odot \mathbf{W}_{hh}\mathbf{h}_{t-1}) + \tanh(\mathbf{W}_{hr}\mathbf{r}_t) \tag{3.6}$$

The reset and update gates thus learn not only the long-term dependency but also the gating information from the dialogue act and the previous hidden state. We call the resulting architecture semantic Refinement GRU with context (RGRU-Context) which is shown in Figure 3.1.

3.1.3 Tying Backward RGRU-Context Model

Due to some sentences may depend on both the past and the future during generation, we train another backward RGRU-Context to utilize the flexibility of the refinement gate $\mathbf{r}_t$, in which we tie its weight matrices such $\mathbf{W}_{rd}$ and $\mathbf{W}_{rh}$ (Equation 3.4) or both. We found that by tying matrix $\mathbf{W}_{rd}$ for both forward and backward RNNs, the proposed generator seems to produce correct and grammatical utterances than those having the only forward RNN. This model called Tying Backward RGRU-Context (TB-RGRU).

3.1.4 Refinement-Adjustment-Output GRU (RAOGRU) Model

Although the *RGRU*-based generators (Tran and Nguyen, 2017b) applying the gating mechanism before general RNN computations show signs of better performance on some NLG domains, it is not clear how the model can prevent undesirable semantic repetitions as the *SCLSTM* model (Wen et al., 2015b) does. Moreover, the *RGRU*-based models treat all input tokens the same at each computational step since the DA vector representation keeps unchanged. This makes them difficult to keep track which slot token has been generated and which one should be remained for next time steps, leading to a high slot error rate ERR.

Despite the improvement over some RNN-based models, the gating-based generators have not been well studied. In this section, we further investigate the gating mechanism-based models in which we propose additional cells into the traditional GRU cell to gate the DA representation.

The proposed model consists of three additional cells: a *Refinement* cell to filter the input tokens (similar to RGRU-Context model), an *Adjustment* cell to control the 1-hot DA vector representation, and an *Output* cell to compute the information which can be outputted together with the GRU output. The resulting architecture called Refinement-Adjustment-Output GRU generator (RAOGRU) demonstrated in Figure 3.2.

Refinement Cell

Inspired by the *refinement gate* of RGRU-Context model, we introduce an additional gate, added before the RNN computation, to filter the input sequence token by token. The refinement gate in Equation 3.4, with the setup to take advantages of capturing the relationship between multiple words, is modified as follows:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_{rd}\mathbf{d}_{t-1} + \mathbf{W}_{rh}\mathbf{h}_{t-1}) \\ \mathbf{x}_t &= \mathbf{r}_t \odot \mathbf{w}_t \end{aligned} \tag{3.7}$$

where $\mathbf{W}_{rd}$ and $\mathbf{W}_{rh}$ are weight matrices, $\odot$ is an element-wise product. The $\odot$ operator plays an important role in word-level matching in which it both learns the vector similarity and reserves information about the two vectors. The new input $\mathbf{x}_t$ contains a combination information of the original input $\mathbf{w}_t$, the dialogue act $\mathbf{d}_{t-1}$, and the context $\mathbf{h}_{t-1}$. Note that while the dialogue act $\mathbf{d}$ of RGRU-Context model stays *unchanged* during sentence processing (Tran and Nguyen, 2017b), it is *adjustable* step by step in this proposed architecture.

GRU Cell

Taking advantages of gating mechanism in LSTM model (Hochreiter and Schmidhuber, 1997) to deal with the gradient exploding problem in RNN, we further apply the refinement gate $\mathbf{r}_t$ deeper into the GRU activation units. The two GRU gates, which are an update gate $\mathbf{z}_t$ to balance between previous activation $\mathbf{h}_{t-1}$ and the candidate activation $\tilde{\mathbf{h}}_t$, and a reset gate $\mathbf{f}_t$ to forget the previous state, are then modified as follows:

$$\begin{aligned} \mathbf{f}_t &= \sigma(\mathbf{W}_{fx}\mathbf{x}_t + \mathbf{W}_{fh}\mathbf{h}_{t-1} + \mathbf{W}_{fr}\mathbf{r}_t) \\ \mathbf{z}_t &= \sigma(\mathbf{W}_{zx}\mathbf{x}_t + \mathbf{W}_{zh}\mathbf{h}_{t-1} + \mathbf{W}_{zr}\mathbf{r}_t) \end{aligned} \tag{3.8}$$

where $\mathbf{W}_{[..]}$ are weight matrices, σ is the sigmoid function, and $\mathbf{f}_t$, $\mathbf{z}_t$ are reset and update gates, respectively. The candidate activation $\tilde{\mathbf{h}}_t$ and the activation $\bar{\mathbf{h}}_t$ are computed as follows:

$$\begin{aligned} \tilde{\mathbf{h}}_t &= \tanh(\mathbf{W}_{hx}\mathbf{x}_t + f_t \odot \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hr}\mathbf{r}_t) \\ \bar{\mathbf{h}}_t &= \mathbf{z}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{z}_t) \odot \tilde{\mathbf{h}}_t \end{aligned} \tag{3.9}$$

where $\mathbf{W}_{hx}$, $\mathbf{W}_{hh}$, and $\mathbf{W}_{hr}$ are weight matrices. Note that while the GRU cell in the previous work (Tran and Nguyen, 2017b) only depended on the *constant* dialogue act representation vector $\mathbf{d}$, the GRU gates and candidate activation in this architecture are modified to depend on the refinement gate $\mathbf{r}_t$. This allows the information flow controlled by the two gating units pass long distance in a sentence.

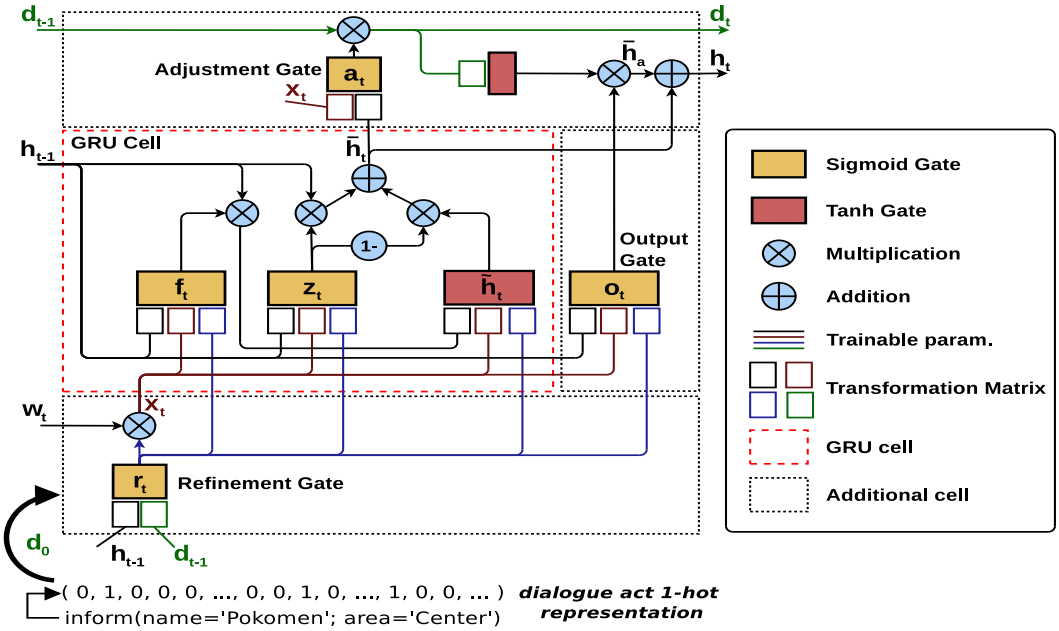


Figure 3.2: RAOGRU cell proposed in this chapter, which consists of four components: a Refinement Cell, a traditional GRU cell, an Adjustment cell, and an Output cell. At each time step, the Refinement cell calculates new input tokens based on a combination of previous DA representation and previous hidden state, the GRU cell mainly in charge of surface realization, and the Adjustment cell and the Output cell compute how much information of the DA vector should be retained for future time steps and those can be contributed to the output.

Adjustment cell

The Refinement gate (Tran and Nguyen, 2017b) showed its ability to filter the input sequences to new inputs which convey useful information before putting into the RNN computation. However, the model treats all input tokens with the same DA vector representation which remains unchanged at every time steps $t = 1, 2, .., T$, where T is length of the input sequence. As a result, the models are difficult to keep track which slot tokens have been generated and which ones should remain for future time steps, leading to a high score of slot error rate ERR. To tackle this problem, inspired by work of (Tran and Nguyen, 2017a) in which an Adjustment cell is introduced on top of the LSTM to gate feature vector $\mathbf{d}_t$, we stack an additional cell on the upper part of the traditional GRU cell.

The additional cell, at time step t, calculates how the output $\bar{\mathbf{h}}_t$ of the traditional GRU affect the control vector $\mathbf{d}_t$ as follows:

$$\begin{aligned} \mathbf{a}_t &= \sigma(\mathbf{W}_{ax}\mathbf{x}_t + \mathbf{W}_{ah}\bar{\mathbf{h}}_t) \\ \mathbf{d}_t &= \mathbf{d}_{t-1} \odot \mathbf{a}_t \end{aligned} \tag{3.10}$$

where $\mathbf{W}_{ax}$ and $\mathbf{W}_{ah}$ are weight matrices, $\mathbf{d}_t$ is a control vector starting with $\mathbf{d}_0$ which is an 1-hot vector representation of given Dialogue Act. Here $\mathbf{W}_{ax}$ and $\mathbf{W}_{ah}$ function as keyword and key-phrase detector which learn to keep track certain patterns of generated tokens associate with certain slots. $\mathbf{a}_t$ called an *Adjustment* gate as its task is to mange what information have been generated by the DA representation and what information should be preserved for next time steps. We propose two models as follows:

RAGRU model

In the first setup, we consider how much of information preserved in the control vector $\mathbf{d}_t$ can be contributed to the model output, in which an additional output is computed by applying the candidate activation $\tilde{\mathbf{h}}_t$ on the remaining information in $\mathbf{d}_t$ as follows:

$$\begin{aligned} \mathbf{c}_a &= \mathbf{W}_{od}\mathbf{d}_t \\ \bar{\mathbf{h}}_a &= \tilde{\mathbf{h}}_t \odot \tanh(\mathbf{c}_a) \end{aligned} \tag{3.11}$$

where $\mathbf{W}_{od}$ is a weight matrix which projects the control vector $\mathbf{d}_t$ into the output space, and $\bar{\mathbf{h}}_a$ is output of the Adjustment cell. The result architecture called Refinement-Adjustment GRU (*RAGRU*) model shown in Figure 3.2.

RAOGRU model

Despite achieving better results, we observed that it might not be sufficient to directly compute the Adjustment output as in Equation (3.11) since the GRU candidate inner activation $\tilde{\mathbf{h}}_t$ may not straightforwardly affect the outer activation $\bar{\mathbf{h}}_a$. Inspired by work of (Hochreiter and Schmidhuber, 1997) in effectively using the gating mechanisms to deal with the exploring or vanishing gradient problems, we propose an additional *Output* gate which acts as the LSTM output gate as follows:

$$\mathbf{o}_t = \sigma(\mathbf{W}_{ox}\mathbf{x}_t + \mathbf{W}_{oh}\mathbf{h}_{t-1} + \mathbf{W}_{or}\mathbf{r}_t) \tag{3.12}$$

where $\mathbf{W}_{ox}$, $\mathbf{W}_{oh}$, and $\mathbf{W}_{or}$ are weight matrices, respectively. Equation (3.11) is then modified to compute the Adjustment output as follows:

$$\begin{aligned} \mathbf{c}_a &= \mathbf{W}_{od}\mathbf{d}_t \\ \bar{\mathbf{h}}_a &= \mathbf{o}_t \odot \tanh(\mathbf{c}_a) \end{aligned} \tag{3.13}$$

where $\mathbf{W}_{od}$ is a weight matrix, $\mathbf{o}_t$ is the proposed Output gate which decides how much of the nonlinear transformation of the control vector $\mathbf{d}_t$ contributes to the output. The result architecture called Refinement-Adjustment-Output GRU (*RAOGRU*) model is depicted in Figure 3.2.

The final output of the network is a combination of both outputs of the traditional GRU cell and the Adjustment cell, which are computed as follows:

$$\mathbf{h}_t = \bar{\mathbf{h}}_t + \bar{\mathbf{h}}_a \tag{3.14}$$

Finally, the output distribution is computed by applying a softmax function g, from which we can sample to obtain the next token:

$$\begin{aligned} P(w_{t+1} \mid w_t, ...w_0, \mathbf{DA}) &= g(\mathbf{W}_{ho}\mathbf{h}_t) \\ w_{t+1} &\sim P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{d}_t) \end{aligned} \tag{3.15}$$

where $\mathbf{W}_{ho}$ is a weight matrix.

3.2 Experiments

We conducted extensive experiments to assess the effectiveness of the proposed models on a variety of datasets and model architectures in order to compare their performance with prior methods.

3.2.1 Experimental Setups

The generators were implemented using the TensorFlow library (Abadi et al., 2016) and trained with a ratio 3:1:1 of training, validation and testing data. The training and decoding procedures are described in Sections 2.5.1 and 2.5.2, respectively. 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. To further understand the effectiveness of the proposed methods we: (i) performed an incremental construction of the proposed models to demonstrate the contribution of each proposed cells (Tables 3.1, 3.2), (ii) trained general models by pooling data from all domains together and tested them in each individual domain (Figure 3.3, and (iii) further conducted experiments to compare the *RGRU-Context* with the *SCLSTM* in a variety of setups on proportion of training corpus, beam size, and top-k best selecting results.

Moving from a limited domain NLG to an open domain NLG raises some problems because of exponentially increasing semantic input elements. Therefore, it is important to build an open domain NLG that can leverage as much of abilities of functioning from existing domains. There have been several work trying to solve this problem, such as (Mrkšić et al., 2015) utilizing RNN-based model for multi-domain dialogue state tracking (Williams, 2013; Gašić et al., 2015) adapting of SDS components to new domains. (Wen et al., 2016a) using a procedure to train multi-domain via multiple adaptation steps, in which a model was trained on counterfeited data by using semantically similar slots from the new domain instead of the slots belonging to the out-of-domain dataset, then fine tune the new domain on the out-of-domain trained model. To examine the model scalability we trained adaptation models on pooling data from Restaurant and Hotel domains, then fine tuned the models on the Laptop domain with varied amount of adaptation data (Figure 3.4),

3.2.2 Evaluation Metrics and Baselines

The generator performances were evaluated using two metrics, BLEU and slot error rate ERR, by adopting code from an NLG toolkit[3]. We compared the proposed models against strong baselines which have been recently published as NLG benchmarks.

- *Gating-based models*, including: HLSTM (Wen et al., 2015a) which uses a heuristic gate to ensure that all of the attribute-value information was accurately captured when generating, SCLSTM (Wen et al., 2015b) which learns the gating signal and language model jointly.

- *Attention-based model* Enc-Dec (Wen et al., 2016b) which applies the attention mechanism to an RNN encoder-decoder by separate computation of slots and values.

3.3 Results and Analysis

We conducted extensive experiments on the proposed models and compared results against previous methods. Overall, the proposed models consistently achieve better performance in comparison with previous gating- and attention-based regarding both evaluation metrics across all domains.

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

Table 3.1: Performance comparison on four datasets in terms of the BLEU and the slot error rate ERR(%) scores. The results were produced by training each network on 5 random initializations and selected model yielded the highest validation BLEU score. The best and second best models highlighted in **bold** and ***italic*** face, respectively.

Model	**Restaurant**		**Hotel**		**Laptop**		**TV**	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
HLSTM	0.7466	0.74%	0.8504	2.67%	0.5134	1.10%	0.5250	2.50%
Enc-Dec	0.7398	2.78%	0.8549	4.69%	0.5108	4.04%	0.5182	3.18%
SCLSTM	0.7525	**0.38%**	0.8482	3.07%	0.5116	***0.79%***	0.5265	2.31%
RGRU-Base	0.7549	0.56%	0.8640	1.21%	0.5190	1.56%	0.5305	1.62%
RGRU-Context	0.7634	0.49%	0.8776	0.98%	0.5191	1.19%	0.5311	1.33%
TB-RGRU	0.7637	***0.47%***	0.8642	1.56%	0.5208	0.93%	0.5312	1.01%
RAGRU	***0.7734***	0.53%	***0.8895***	***0.46%***	***0.5216***	0.85%	***0.5347***	***0.63%***
RAOGRU	**0.7762**	**0.38%**	**0.8907**	**0.17%**	**0.5227**	**0.47%**	**0.5387**	**0.60%**

Table 3.2: Performance comparison of the proposed models on four datasets in terms of the BLEU and the error rate ERR(%) scores. The results were averaged over 5 random initialization networks. The best and second best models highlighted in **bold** and ***italic*** face, respectively.

Model	**Restaurant**		**Hotel**		**Laptop**		**TV**	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
HLSTM	0.7436	0.85%	0.8488	2.79%	0.5130	1.15%	0.5240	2.65%
Enc-Dec	0.7358	2.98%	0.8537	4.78%	0.5101	4.24%	0.5142	3.38%
SCLSTM	0.7543	***0.57%***	0.8469	3.12%	0.5109	***0.89%***	0.5235	2.41%
RGRU-Base	0.7526	1.33%	0.8622	1.12%	0.5165	1.79%	0.5311	1.56%
RGRU-Context	0.7614	0.99%	0.8677	1.75%	0.5182	1.41%	0.5312	1.37%
TB-RGRU	0.7608	0.88%	0.8584	1.63%	0.5188	1.35%	0.5316	1.27%
RAGRU	***0.7729***	0.64%	***0.8887***	***0.60%***	***0.5206***	0.92%	***0.5341***	***0.82%***
RAOGRU	**0.7730**	**0.49%**	**0.8903**	**0.41%**	**0.5228**	**0.60%**	**0.5368**	**0.72%**

3.3.1 Model Comparison in Individual Domain

The incremental construction studies (Tables 3.1, 3.2) demonstrate the contribution of different model components in which the models were assessed as a base model (*RGRU-Context*), with Adjustment cell (*RAGRU*), and with Output cell (*RAOGRU*). A comparison between the gating-based models clearly shows that the Adjustment cell contributes to reducing the slot error rate ERR score since it can effectively prevent the undesirable slot repetitions by gating the DA vector, while the additional Output cell provides an improved performance on both evaluation metrics across all domains since it can separate the information outputs from the traditional GRU cell and the Adjustment cell.

Moreover, Table 3.2 further demonstrates the stable strength of the proposed models since the proposed models not only outperform the gating-based models (*RGRU-Base*, *RGRU-Context*, *TB-RGRU*) and but also show significant improved results over the attention-based model (*End-Dec*) by a large margin. A comparison of the two proposed generators (RGRU-Context and TB-RGRU) is also shown in Table 3.2. Without the backward RNN reranker, the RGRU-Context generator seems to have worse performance with a higher score of slot error rate ERR. However, using the backward RGRU reranker can improve the results in both evaluation metrics. This reranker provides benefit to the generator to produce higher-quality utterances.

These demonstrate the importance of the proposed components: the Refinement cell in filtering input information, the Adjustment cell in controlling the feature vector (see Examples in Figure 3.7), and the Output cell in calculating the additional output.

3.3.2 General Models

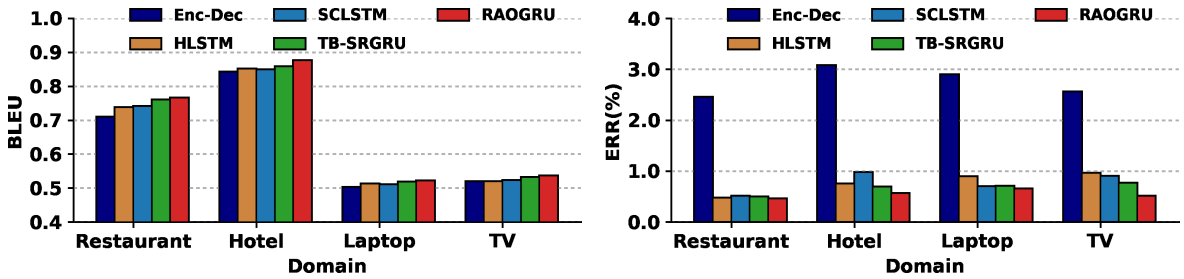


Figure 3.3: Gating-based generators comparison of the general models on four different domains.

Figure 3.3 shows a comparison performance of general models as described in Section 3.2.1. The results are consistent with those in Table 3.1, 3.2, in which the *RAOGRU* again has better performance than the gating-based models (*HLSTM*, *SCLSTM*,*RGRU-Base*, and *TB-RGRU*) across all domains in terms of the BLEU and the ERR scores. In comparison between attention-based models (Figure 3.3), while the *Enc-Dec* has difficulty in reducing the ERR score, the models *RAOGRU* has best results regarding the BLEU score and effectively drive down the slot error rate ERR score since it can control the feature vector DA to prevent the undesirable slot repetitions.

These indicate the relevant contribution of the proposed components which are the Refinement, Adjustment and Output cells to the original architecture. The Refinement gate can effectively select the beneficial input information before putting them into the traditional GRU cell, while the Adjustment and Output cells with gating DA vector can effectively control the information flow during generation.

3.3.3 Adaptation Models

Figure 3.4 shows domain scalability of the three models in which the models were first trained by pooling out-of-domain Restaurant and Hotel datasets together. The models were then fine-tuned the parameters with different proportion of in-domain training data (Laptop domain). The proposed model *RAOGRU* again outperforms both previous models (*Enc-Dec*, *SCLSTM*) in both cases where the sufficient in-domain data is used (as in Figure 3.4-*left*) and the limited in-domain data is fed (Figure 3.4-*right*). The Figure 3.4-*right* also indicates that the *RAOGRU* model can adapt to a new, unseen domain faster than the previous models.

3.3.4 Model Comparison on Tuning Parameters

Figure 3.5 shows a comparison of two generators trained with different proportion of data evaluated on two metrics. As can be seen in Figure 3.5a, the SCLSTM model achieves better results than RGRU-Context model on both of BLEU and ERR scores since a small amount of training

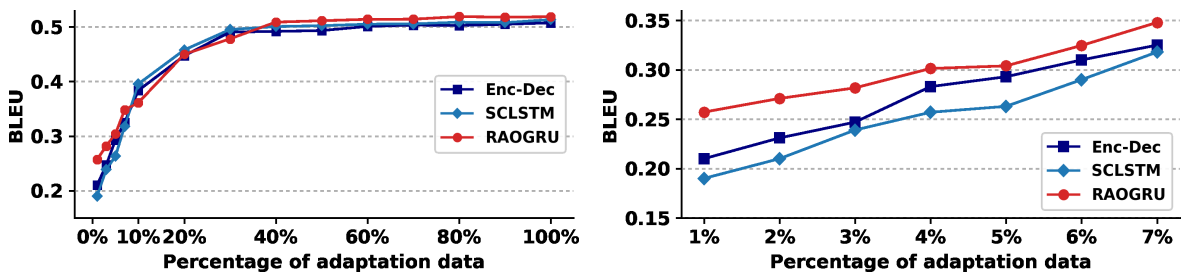
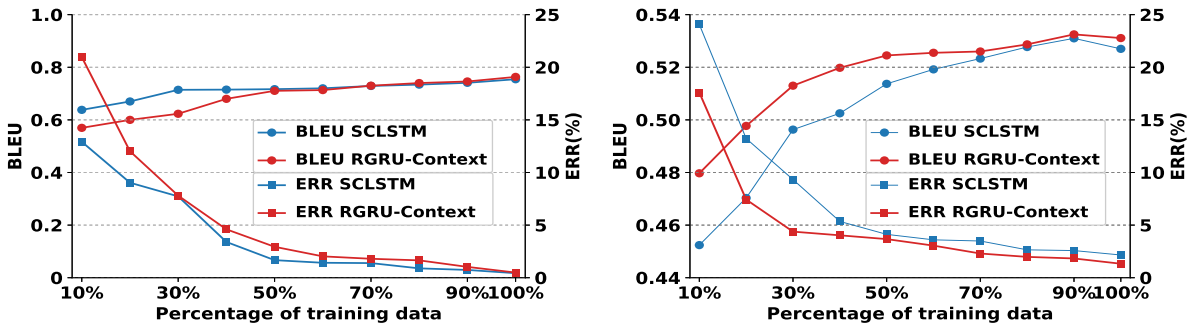


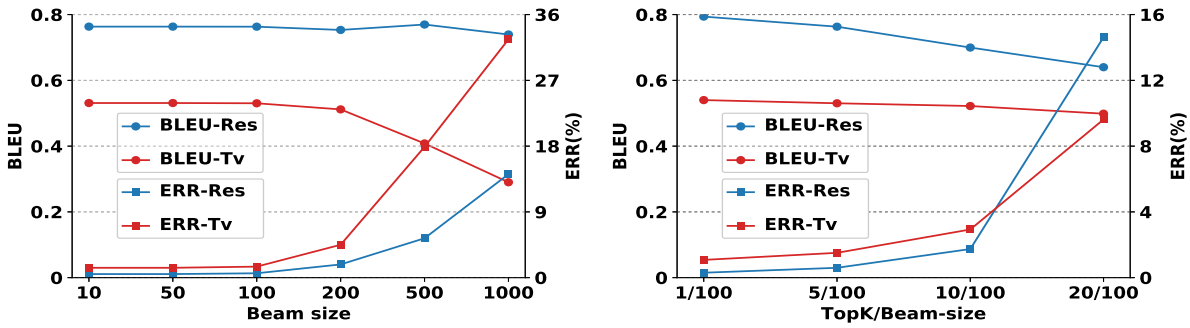
Figure 3.4: Performance on Laptop domain with varied amount of the adaptation training data when adapting models trained on Restaurant+Hotel.



(a) Curves on the Restaurant dataset

(b) Curves on the TV dataset

Figure 3.5: Comparison of two generators RGRU-Context and SCLSTM which are trained with different proportion of training data.



(a) Curves on variation of Beam size

(b) Curves on variation of Top-k/Beam-size

Figure 3.6: RGRU-Context generator was trained with different Beam size (a) and Top-k best results (b) and evaluated on Restaurant and TV datasets.

data was provided. However, the RGRU-Context obtains the higher BLEU score and slightly higher ERR score as more training data was fed. On the other hand, in a more diverse dataset TV, the RGRU-Context model consistently outperforms the SCLSTM on both evaluation metrics no matter how much training data is (Figure 3.5b). The reason is mainly due to the ability of refinement gate which feeds to the GRU model a new input $\mathbf{x}_t$ conveying useful information filtered from the original input and the gating mechanism; this gate also keeps the pattern of the generated utterance during generation. As a result, a better realization of unseen slot-value pairs is obtained.

Figure 3.6a shows an effect of beam size on the RGRU-Context model evaluated on Restaurant and TV datasets. As can be seen, the models perform worse in terms of degrading the BLEU score and upgrading the slot error rate ERR score when the beam size increases. The model seems to perform best with beam size less than 100. Figure 3.6b presents an effect of top-k best results in which we fixed the beam size at 100 and top-k best results varied as k = 1, 5, 10 and 20. In each case, the BLEU and the error rate ERR scores were computed on Restaurant and TV datasets. The results are consistent with Figure 3.6a in which the BLEU and ERR scores get worse as more top-k best utterances were chosen.

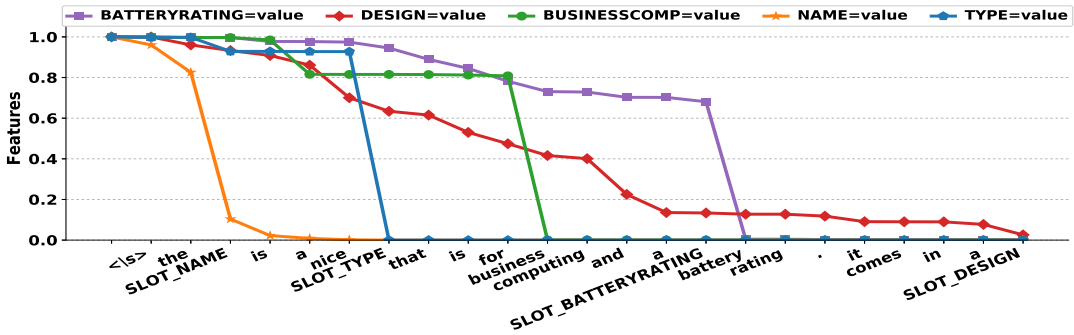


Figure 3.7: Example showing how RAOGRU control the DA feature value vector $\mathbf{d}_t$. The model generally shows its ability to detect words and phases describing a corresponding slot-value pair

3.3.5 Model Comparison on Generated Utterances

Table 3.3: Comparison of top responses generated for different models. Errors are marked in color ([missing], misplaced information). [OK] denotes successful generation.

Model	Generated Responses from Laptop Domain
Input DA	?compare(name='*satellite tartarus 56*'; platform='*windows 8*'; dimension='*15.3 inch*'; name='*satellite achelous 45*'; platform='*windows 8.1*'; dimension='*15.5 inch*')
Reference	the *satellite tartarus 56* has a *15.3 inch* dimension and uses *windows 8*, whereas *satellite achelous 45* has a *15.5 inch* dimension and a *windows 8.1* platform. which one do you prefer
HLSTM	the *satellite tartarus 56* is *15.3 inch* -s and operates on *windows 8* and the *satellite achelous 45* has a *15.5 inch* display. which one do you want [windows 8.1]
Enc-Dec	the *satellite tartarus 56* is *15.3 inch*. the *satellite achelous 45* operates on *windows 8* and has a *15.5 inch* screen . which one do you prefer [windows 8.1]
SCLSTM	the *satellite tartarus 56* operates on *windows 8* and has a *15.3 inch* display and is *15.5 inch* -s. which one do you prefer [satellite achelous 45, windows 8.1]
RGRU-Context	the *satellite tartarus 56* has a *15.3 inch* dimension. the *satellite achelous 45* is *15.5 inch* -s . which one do you want [windows 8, windows 8.1]
TB-RGRU	the *satellite tartarus 56* is *15.3 inch* -s and runs on *windows 8*. the *satellite achelous 45* is *15.5 inch* -s. which one do you prefer [windows 8.1]
RAGRU	the *satellite tartarus 56* has a *15.3 inch* screen, the *satellite achelous 45* is *15.5 inch* and runs *windows 8* . which one do you prefer [windows 8.1]
RAOGRU	the *satellite tartarus 56* has a *15.3 inch* dimension and operates on *windows 8*. the *satellite achelous 45* operates on *windows 8.1* and is *15.5 inch* -s. which one do you prefer [OK]

A comparison of top generated responses for given DA between different models is shown in Table 3.3. While the previous models still produce some errors (missing and misplaced information), the proposed model (*RAOGRU*) can generate appropriate sentences. Figure 3.7 also

demonstrate how the feature vector DA is controlled during generation, in which the *RAOGRU* model can generally drive down the DA feature.

3.4 Conclusion

This chapter have presented *gating*-based neural language generators for SDSs, in which three additional cells (*Refinement*, *Adjustment*, and *Output* cells) are introduced to select, control the semantic elements, and generate the required sentence. We assessed the proposed models on four different NLG domains and compared those against previous generators. The proposed models empirically show consistent improvement over the previous gating-based model on both BLEU and ERR evaluation metrics. The gating-based generators mostly address NLG problems of adequacy, completeness and adaptability, in which the models showed ability to handle special slots, such as binary slots, slots that take *dont_care* value, as well as to effectively avoid slots repetition by controlling the feature vector DA. The proposed gating-based models also showed signs of adaptability to quickly scale to a new, unseen domain no matter how much the training in-domain data was fed. In the next chapter 4, we continue to improve the gating-based models by integrating the models into a unified sequence to sequence model (Vinyals and Le, 2015) with an effective attention mechanism (Bahdanau et al., 2014).

Chapter 4

Hybrid based NLG

In this chapter, we present a *hybrid* NLG framework that leverages the strength of both *gating* and *attention* mechanisms into an RNN Encoder-Decoder (ARED) to tackle the NLG problems of the *adequacy*, *completeness*, and *adaptability*.

As mentioned, current SDSs typically rely on a well-defined ontology for specific domains, such as finding hotel, restaurant, or buying a laptop, television, and so forth, which requires an extremely expensive and time-consuming process for the data collecting. Our *gating*-based generators proposed in Chapter 3 and an ENCDEC Wen et al. (2016b) model showed a sign in domain *scalability* when a limited amount of data is available. However, the goal of building an open domain SDS which can talk about any topic is still a difficult task. Therefore, it is crucial to building an open domain dialogue system that can make as much use of existing abilities of knowledge from other domains or learn from multi-domain datasets.

On the other hand, despite the advantages of *gating*-based mechanism in tackling the NLG problems, previous RNN-based models still have some drawbacks. *First*, none of these models significantly outperform the others in solving NLG problems which have remained unsolved. While the HLSTM (Wen et al., 2015a) cannot handle cases, such as the binary slots and *don't_care* slots, in which these slots cannot be directly delexicalized, the ENCDEC model has difficulty to prevent undesirable semantic repetitions during generation. *Second*, although the SCLSTM, RAOGRU model can generally drive down the feature vector DA (see Figure 3.7), leading to a low score of slot error rate ERR, none of the existing models show significant advantage from out-of-domain data. Furthermore, while the SCLSTM model is limited to generalize to the unseen domains, there are still some generation cases which consist of consecutive slots the RAOGRU cannot fully control the feature vector DA (see Figure 4.1).

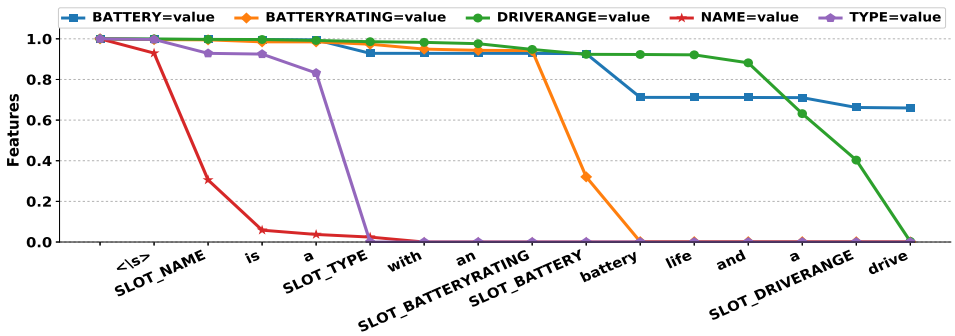


Figure 4.1: RAOGRU may not fully control the feature value vector DA in some generation cases that consist of *consecutive* slots, *e.g.*, SLOT_BATTERYRATING and SLOT_BATTERY.

To deal with the above issues, we first present a novel ARED-based generation framework in Section 4.1, which mainly consists of three components: an Encoder, an Aligner, and a Decoder. We then propose a first novel model, Encoder-Aggregator-Decoder in Section 4.2, an extension of the ARED generation framework. The Aggregator component has two subcomponents: (i) an Aligner which computes the attention over the input sequence, and (ii) a Refiner which are another attention or gating mechanisms to further control and select the semantic elements. This model is based on our work in (Tran et al., 2017a). Lastly, we propose a second novel architecture in Section 4.3, in which a Refinement Adjustment LSTM-based component (RALSTM) is introduced at the decoder side to select, aggregate, and control the semantic information. The second methods yield state-of-the-art results across four NLG domains in terms of the BLEU and slot error rate ERR scores. We publish this work in (Tran and Nguyen, 2017a, 2018c). The results also showed that both proposed generators could quickly adapt to new domains by leveraging the out-of-domain data.

To sum up, we make four key contributions in this chapter:

- Firstly, we present a new general neural language generation framework from which we propose two novel models, Encoder-Aggregator-Decoder and RALSTM generators.
- Secondly, we present a semantic component called Aggregator which is easily integrated into existing ARED-based architecture with several different choices of attention and gating mechanisms, resulting in an end-to-end generator that empirically improved performance over the previous approaches.
- Thirdly, we present an LSTM-based component called RALSTM cell applied on the decoder side of an ARED architecture, resulting in an end-to-end generator that empirically shows significantly improved performances compared with the previous methods.
- Finally, we extensively conducted the experiments to evaluate the models training on varied scenarios, in which we: (i) trained the generators from scratch on each in-domain dataset, including new, unseen domains, (ii) trained general models by pooling all available datasets together then tested them in each domain, (iii) trained adapting models by incremental pooling out-of-domain datasets together, then fine-tuned the trained model by using a limited amount of in-domain data.

4.1 The Neural Language Generator

The neural language generation framework in this study is based on a neural net language generator (Wen et al., 2016b), which consists of three main components: an Encoder to incorporate the target meaning representation as the model inputs, an Aligner to align and control the semantic elements, and a Decoder to generate output utterances. The ARED NLG framework is depicted in Figure 4.2. The Encoder first encodes the meaning representation into input semantic elements which are then aggregated and selected by utilizing an attention-based mechanism by the Aligner. The RNN Decoder at each time step takes as input a 1-hot encoding of a token[1] $\mathbf{w}_t$ and an attentive DA representation $\mathbf{d}_t$. The output of RNN decoder represents the probability distribution of the next token given the previous token, the dialogue act representation, and the current hidden state. At generation time, we can sample from this conditional distribution to

[1]Input texts are delexicalized where slot values are replaced by its corresponding slot tokens.

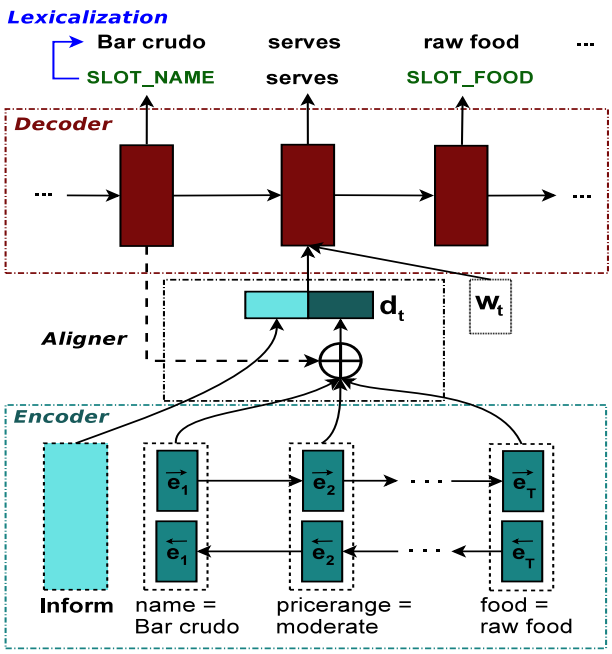


Figure 4.2: Unfold presentation of the ARED neural language generation framework. While the Encoder part is subject to various designs with a BiRNN, the Aligner is an attention mechanism, and the Decoder is typically an RNN network.

obtain the next token in a generated sentence, and feed it as the next input to the RNN Decoder. This process finishes when an end sign is generated (Karpathy and Fei-Fei, 2015), or some constraints are reached (Zhang and Lapata, 2014). The model can produce a sequence of tokens which can finally be lexicalized[2] to form the required utterance.

4.1.1 Encoder

The slots and values are separated parameters used on the encoder side. This embeds the source information into a vector representation $\mathbf{z}_i$ which is a concatenation of embedding vector representation of each slot-value pair and is computed by:

$$\mathbf{z}_i = \mathbf{u}_i \oplus \mathbf{v}_i \tag{4.1}$$

where $\mathbf{u}_i$, $\mathbf{v}_i$ are the i-th slot and value embedding vectors, respectively, and $\oplus$ is vector concatenation. The i index runs over the L given slot-value pairs. In this work, we use a 1-layer, Bidirectional RNN (Bi-RNN) to encode the sequence of slot-value pairs[3] embedding. The Bi-RNN consists of forward and backward RNNs which read the sequence of slot-value pairs from left-to-right and right-to-left to produce forward and backward sequence of hidden states $(\overrightarrow{\mathbf{e}_1}, .., \overrightarrow{\mathbf{e}_L})$, and $(\overleftarrow{\mathbf{e}_1}, .., \overleftarrow{\mathbf{e}_L})$, respectively. We then obtain the sequence of encoded hidden states $\mathbf{E} = (\mathbf{e}_1, \mathbf{e}_2, .., \mathbf{e}_L)$ where $\mathbf{e}_i$ is a sum of the forward hidden state $\overrightarrow{\mathbf{e}_i}$ and the backward one $\overleftarrow{\mathbf{e}_i}$ as follows:

$$\mathbf{e}_i = \overrightarrow{\mathbf{e}_i} + \overleftarrow{\mathbf{e}_i} \tag{4.2}$$

[2]The process in which slot token is replaced by its value.

[3]We treated the set of slot-value pairs as a sequence and use the order specified by slot's name (e.g., slot *address* comes first, *food* follows *address*). We have tried treating slot-value pairs as a set with natural order as in the given DAs. However, this yielded even worse results.

4.1.2 Aligner

The Aligner utilizes an attention mechanism to calculate the DA representation as follows:

$$\beta_{t,i} = \frac{\exp e_{t,i}}{\sum_j \exp e_{t,j}} \tag{4.3}$$

where

$$e_{t,i} = a(\mathbf{e}_i, \mathbf{h}_{t-1}) \tag{4.4}$$

and $\beta_{t,i}$ is the weight of *i*-th slot-value pair calculated by the attention mechanism. The alignment model a is computed by:

$$a(\mathbf{e}_i, \mathbf{h}_{t-1}) = \mathbf{v}_a^\top \tanh(\mathbf{W}_a \mathbf{e}_i + \mathbf{U}_a \mathbf{h}_{t-1}) \tag{4.5}$$

where $\mathbf{v}_a, \mathbf{W}_a, \mathbf{U}_a$ are the weight matrices to learn. Finally, the Aligner calculates dialogue act embedding $\mathbf{d}_t$ as follows:

$$\mathbf{d}_t = \mathbf{a} \oplus \sum_i \beta_{t,i} \mathbf{e}_i \tag{4.6}$$

where **a** is vector embedding of the action type.

4.1.3 Decoder

The decoder of the general neural language generator is typically an RNN. While the RNN decoder takes as input at each time step a 1-hot encoding of a token and the attentive input vector, the output of RNN decoder represents the probability distribution of the next token given the previous token, the dialogue act representation, and the current hidden state.

4.2 The Encoder-Aggregator-Decoder model

In this section, we present our first proposed model, Encoder-Aggregator-Decoder, an extension of the ARED architecture, in which the proposed Aggregator has two main components: (i) an Aligner which computes the attention over the input sequence (as described in section 4.1.2), and (ii) a Refiner which are another attention or gating mechanisms to further select and aggregate the semantic elements. The model architecture is shown in Figure 4.3. To sum up, we make two key contributions in this model:

- We present a semantic component called Aggregator which can be easily integrated into existing (attentive) RNN Encoder-Decoder architecture, resulting in an end-to-end generator that empirically improved performance in comparison with the previous approaches.
- We present several different choices of attention and gating mechanisms which can be effectively applied to the proposed semantic Aggregator.

4.2.1 Gated Recurrent Unit

The encoder and decoder of the proposed model utilize a Gated Recurrent Unit (GRU) network proposed by (Bahdanau et al., 2014), which maps an input sequence $\mathbf{W} = [\mathbf{w}_1, \mathbf{w}_2, .., \mathbf{w}_T]$ to a

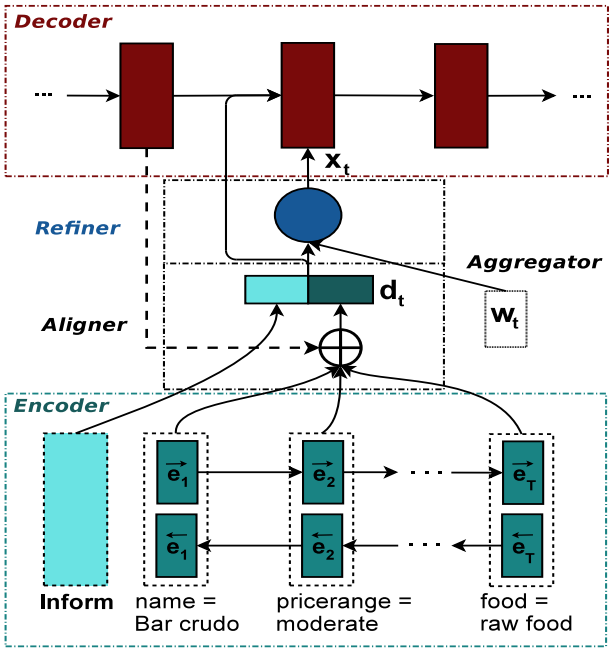


Figure 4.3: The RNN Encoder-Aggregator-Decoder for NLG proposed in this chapter. The output side is an RNN network while the input side is a DA embedding with aggregation mechanism. The Aggregator consists of two parts: an Aligner and a Refiner. The lower part Aligner is an attention over the DA representation calculated by a BiGRU network. Note that the action type embedding $\mathbf{a}$ is not included in the attention mechanism since its task is controlling the style of the sentence. The higher part Refiner computes the new input token $\mathbf{x}_t$ based on the original input token $\mathbf{w}_t$ and the dialogue act attention $\mathbf{d}_t$. There are several choices for Refiner, i.e., gating mechanism or attention mechanism.

sequence of states $\mathbf{H} = [\mathbf{h}_1, \mathbf{h}_2, .., \mathbf{h}_T]$ as follows:

$$\begin{aligned}
\mathbf{r}_i &= \sigma(\mathbf{W}_{rw}\mathbf{w}_i + \mathbf{W}_{rh}\mathbf{h}_{i-1}) \\
\mathbf{u}_i &= \sigma(\mathbf{W}_{uw}\mathbf{w}_i + \mathbf{W}_{uh}\mathbf{h}_{i-1}) \\
\tilde{\mathbf{h}}_i &= \tanh(\mathbf{W}_{hw}\mathbf{w}_i + \mathbf{r}_i \odot \mathbf{W}_{hh}\mathbf{h}_{i-1}) \\
\mathbf{h}_i &= \mathbf{u}_i \odot \mathbf{h}_{i-1} + (1 - \mathbf{u}_i) \odot \tilde{\mathbf{h}}_i
\end{aligned} \tag{4.7}$$

where $\odot$ denotes the element-wise multiplication, $\mathbf{r}_i$ and $\mathbf{u}_i$ are called the reset and update gates respectively, and $\tilde{\mathbf{h}}_i$ is the candidate activation.

4.2.2 Aggregator

The Aggregator consists of two components: an Aligner and a Refiner. The Aligner computes the dialogue act representation (as described in section 4.1.2) while the choices for Refiner can be varied. The Refiner calculates the new input $\mathbf{x}_t$ based on the original input token $\mathbf{w}_t$ and the DA representation. There are several choices to formulate the Refiner such as gating mechanism or attention mechanism. For each input token $\mathbf{w}_t$, the selected mechanism module computes the new input $\mathbf{x}_t$ based on the dialog act representation $\mathbf{d}_t$ and the input token embedding $\mathbf{w}_t$, and is formulated by:

$$\mathbf{x}_t = f_R(\mathbf{d}_t, \mathbf{w}_t) \tag{4.8}$$

where f_R is a refinement function, in which each input token is refined (or filtered) by the dialogue act attention information before putting into the RNN decoder. By this way, we can represent the whole sentence based on this refined input using RNN model.

Attention Mechanism: Inspired by work of (Cui et al., 2016), in which an attention-over-attention was introduced in solving reading comprehension tasks, we place another attention applied for Refiner over the attentive Aligner, resulting in a model Attentional Refiner over Attention (ARoA).

- ARoA with Vector (ARoA-V): We use a simple attention where each input token representation is weighted according to dialogue act attention as follows:

$$\begin{aligned} \beta_t &= \sigma(\mathbf{V}_{ra}^{\top}\mathbf{d}_t) \\ f_R(\mathbf{d}_t, \mathbf{w}_t) &= \beta_t * \mathbf{w}_t \end{aligned} \tag{4.9}$$

  where $\mathbf{V}_{ra}$ is a refinement attention vector which is used to determine the dialogue act attention strength, and σ is sigmoid function to normalize the weight β_t between 0 and 1.

- ARoA with Matrix (ARoA-M): ARoA-V uses only a vector $\mathbf{V}_{ra}$ to weight the DA attention. It may be better to use a matrix to control the attention information. The Equation 4.9 is modified as follows:

$$\begin{aligned} \mathbf{V}_{ra} &= \mathbf{W}_{aw}\mathbf{w}_t \\ \beta_t &= \sigma(\mathbf{V}_{ra}^{\top}\mathbf{d}_t) \\ f_R(\mathbf{d}_t, \mathbf{w}_t) &= \beta_t * \mathbf{w}_t \end{aligned} \tag{4.10}$$

  where $\mathbf{W}_{aw}$ is a refinement attention matrix.

- ARoA with Context (ARoA-C): The attention in ARoA-V and ARoA-M may not capture the relationship between multiple tokens. In order to add context information into the attention process, we modify the attention weights in Equation 4.10 with additional history information $\mathbf{h}_{t-1}$:

$$\begin{aligned} \mathbf{V}_{ra} &= \mathbf{W}_{aw}\mathbf{w}_t + \mathbf{W}_{ah}\mathbf{h}_{t-1} \\ \beta_t &= \sigma(\mathbf{V}_{ra}^{\top}\mathbf{d}_t) \\ f_R(\mathbf{d}_t, \mathbf{w}_t, \mathbf{h}_{t-1}) &= \beta_t * \mathbf{w}_t \end{aligned} \tag{4.11}$$

  where $\mathbf{W}_{aw}, \mathbf{W}_{ah}$ are parameters to learn, $\mathbf{V}_{ra}$ is the refinement attention vector same as above, which contains both DA attention and context information.

Gating Mechanism: We present two gating-based models by simply using element-wise operators (multiplication or addition) to gate the information between the two vectors $\mathbf{d}_t$ and $\mathbf{w}_t$ as follows:

- Multiplication (GR-MUL): The element-wise multiplication plays a part in word-level matching which learns not only the vector similarity, but also preserve information about the two vectors:

$$f_R(\mathbf{d}_t, \mathbf{w}_t) = \mathbf{W}_{gd}\mathbf{d}_t \odot \mathbf{w}_t \tag{4.12}$$

- Addition (GR-ADD):

$$f_R(\mathbf{d}_t, \mathbf{w}_t) = \mathbf{W}_{gd}\mathbf{d}_t + \mathbf{w}_t \tag{4.13}$$

4.2.3 Decoder

The decoder uses a conventional GRU model as described in Section 4.2.1. In this work, we propose to apply the DA representation and the refined inputs deeper into the GRU cell. Firstly, the GRU reset and update gates can be further influenced on the DA attentive information $\mathbf{d}_t$. The calculation for the reset and update gates are modified as follows:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_{rx}\mathbf{x}_t + \mathbf{W}_{rh}\mathbf{h}_{t-1} + \mathbf{W}_{rd}\mathbf{d}_t) \\ \mathbf{u}_t &= \sigma(\mathbf{W}_{ux}\mathbf{x}_t + \mathbf{W}_{uh}\mathbf{h}_{t-1} + \mathbf{W}_{ud}\mathbf{d}_t) \end{aligned} \tag{4.14}$$

where $\mathbf{W}_{rd}$ and $\mathbf{W}_{ud}$ act like background detectors that learn to control the style of the generating sentence. Secondly, the candidate activation $\tilde{\mathbf{h}_t}$ is also modified to depend on the DA representation as follows:

$$\tilde{\mathbf{h}_t} = \tanh(\mathbf{W}_{hx}\mathbf{x}_t + \mathbf{r}_t \odot \mathbf{W}_{hh}\mathbf{h}_{t-1} + \mathbf{W}_{hd}\mathbf{d}_t) + \tanh(\mathbf{W}_{dc}\mathbf{d}_t) \tag{4.15}$$

The hidden state is then computed by:

$$\mathbf{h}_t = \mathbf{u}_t \odot \mathbf{h}_{t-1} + (1 - \mathbf{u}_t) \odot \tilde{\mathbf{h}_t} \tag{4.16}$$

Finally, the output distribution is computed by applying a softmax function g, and the distribution is sampled to obtain the next token,

$$\begin{aligned} P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{z}) &= g(\mathbf{W}_{ho}\mathbf{h}_t) \\ w_{t+1} &\sim P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{z}) \end{aligned} \tag{4.17}$$

4.3 The Refinement-Adjustment-LSTM model

In this section, we present our second proposed model, Refinement-Adjustment-LSTM (RALSTM), an extension of the ARED framework, which consists of three main components: (i) an Encoder that incorporates the target meaning representation (MR) as the model inputs, (ii) an Aligner that aligns and controls the semantic elements, and (iii) an RNN Decoder that generates output sentences. The RALSTM generator architecture is shown in Figure 4.4. The Encoder first encodes the MR into input semantic elements which are then aggregated and selected by utilizing an attention-based mechanism by the Aligner. The input to the RNN Decoder at each time step is a 1-hot encoding of a token $\mathbf{w}_t$ and an attentive DA representation $\mathbf{d}_t$. At each time step t, RNN Decoder also computes how much the feature value vector $\mathbf{s}_{t-1}$ retained for the next computational steps, and adds this information to the RNN output which represents the probability distribution of the next token $\mathbf{w}_{t+1}$.

To sum up, we make three key contributions in this proposed model:

- We present an LSTM-based cell called RALSTM applied on the decoder side of an ARED model, resulting in an end-to-end generator that empirically shows significantly improved performances in comparison with the previous approaches.
- We extensively conduct the experiments to evaluate the models training from scratch on each in-domain dataset.
- We empirically assess the models' ability to: learn from multi-domain datasets by pooling all available training datasets; and adapt to a new, unseen domain by feeding a limited amount of in-domain data.

In the following sections, we present in detail each components of the proposed RALSTM generator.

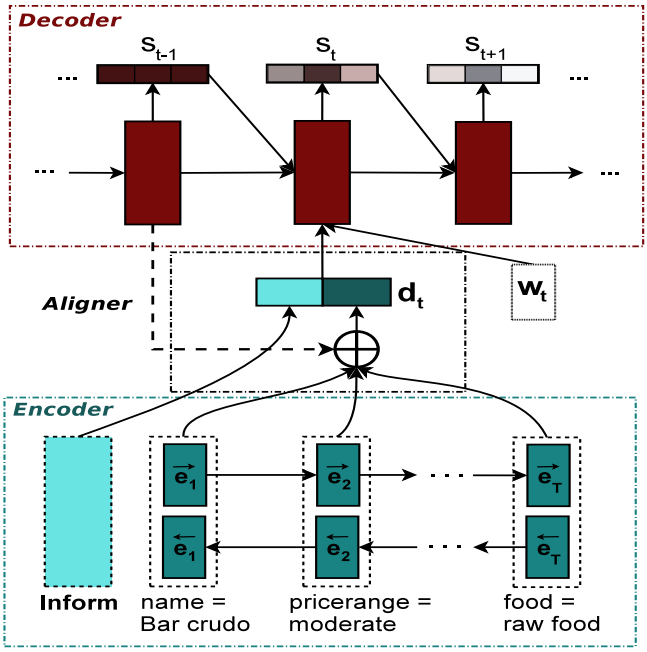


Figure 4.4: Unrolled presentation of the ARED-based neural language generator. The Encoder part is a BiLSTM, the Aligner is an attention mechanism over the encoded inputs, and the Decoder is the proposed RALSTM model conditioned on a 1-hot representation vector **s**. The fading color of the vector **s** indicates retaining information for future computational time steps.

4.3.1 Long Short Term Memory

The encoder and decoder of the RALSTM model utilize a Long Short Term Memory (LSTM) network proposed by (Hochreiter and Schmidhuber, 1997) in which the input gate $\mathbf{i}_i$, forget gate $\mathbf{f}_t$ and output gate $\mathbf{o}_t$ are introduced to control information flow and computed as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W} \begin{pmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{pmatrix} \tag{4.18}$$

where $\mathbf{W}$ is model parameters. The memory cell value $\mathbf{c}_t$ is computed as follows:

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t \\ \mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \end{aligned} \tag{4.19}$$

where $\mathbf{h}_t$ is the output.

4.3.2 RALSTM Decoder

The proposed RALSTM cell, which is demonstrated in Figure 4.5, applied for Decoder side consists of three components: a Refinement cell, a traditional LSTM cell, and an Adjustment cell:

Firstly, instead of feeding the original input token $\mathbf{w}_t$ into the RNN cell, the input is recomputed by using a semantic gate as follows:

$$\begin{aligned} \mathbf{r}_t &= \sigma(\mathbf{W}_{rd}\mathbf{d}_t + \mathbf{W}_{rh}\mathbf{h}_{t-1}) \\ \mathbf{x}_t &= \mathbf{r}_t \odot \mathbf{w}_t \end{aligned} \tag{4.20}$$

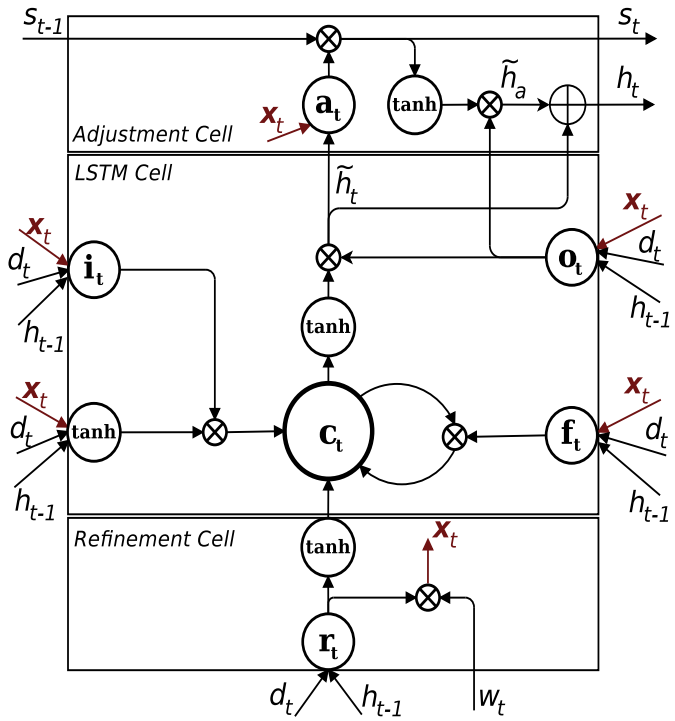


Figure 4.5: The RALSTM cell proposed in this work, which consists of three components: a Refinement cell, a traditional LSTM cell, and an Adjustment cell. At time step t, while the Refinement cell computes new input token $\mathbf{x}_t$ based on the original input token and the attentional DA representation $\mathbf{d}_t$, the Adjustment cell calculates how much information of the slot-value pairs can be generated by the LSTM cell.

where $\mathbf{W}_{rd}$ and $\mathbf{W}_{rh}$ are weight matrices. Element-wise multiplication $\odot$ plays a part in word-level matching which not only learns the vector similarity but also preserves information about the two vectors. $\mathbf{W}_{rh}$ acts like a key phrase detector that learns to capture the pattern of generation tokens or the relationship between multiple tokens. In other words, the new input $\mathbf{x}_t$ consists of information of the original input token $\mathbf{w}_t$, the DA representation $\mathbf{d}_t$, and the hidden context $\mathbf{h}_{t-1}$. $\mathbf{r}_t$ is called a Refinement gate because the input tokens are refined by a combination gating information of the attentive DA representation $\mathbf{d}_t$ and the previous hidden state $\mathbf{h}_{t-1}$. By this way, we can represent the whole sentence based on the refined inputs.

Secondly, the traditional LSTM network proposed by (Hochreiter and Schmidhuber, 1997) in which the input gate $\mathbf{i}_i$, forget gate $\mathbf{f}_t$ and output gates $\mathbf{o}_t$ are introduced to control information flow and computed as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W}_{4n,4n} \begin{pmatrix} \mathbf{x}_t \\ \mathbf{d}_t \\ \mathbf{h}_{t-1} \end{pmatrix} \tag{4.21}$$

where n is hidden layer size, $\mathbf{W}_{4n,4n}$ is model parameters. The cell memory value $\mathbf{c}_t$ is modified to depend on the DA representation as:

$$\begin{aligned} \mathbf{c}_t &= \mathbf{f}_t \odot \mathbf{c}_{t-1} + \mathbf{i}_t \odot \hat{\mathbf{c}}_t + \tanh(\mathbf{W}_{cr}\mathbf{r}_t) \\ \tilde{\mathbf{h}}_t &= \mathbf{o}_t \odot \tanh(\mathbf{c}_t) \end{aligned} \tag{4.22}$$

where $\tilde{\mathbf{h}}_t$ is the output.

Thirdly, inspired by work of (Wen et al., 2015b) in which the generator was further conditioned on a 1-hot representation vector $\mathbf{s}$ of given dialogue act, and work of (Lu et al., 2016) that proposed a visual sentinel gate to make a decision on whether the model should attend to the image or the sentinel gate, an additional gating cell is introduced on top of the traditional LSTM to gate another controlling vector $\mathbf{s}$. Figure 4.8 shows how RALSTM controls the DA vector $\mathbf{s}$. First, starting from the 1-hot vector of the DA $\mathbf{s}_0$, at each time step t the proposed cell computes how much the LSTM output $\tilde{\mathbf{h}}_t$ affects the DA vector, which is computed as follows:

$$\begin{aligned} \mathbf{a}_t &= \sigma(\mathbf{W}_{ax}\mathbf{x}_t + \mathbf{W}_{ah}\tilde{\mathbf{h}}_t) \\ \mathbf{s}_t &= \mathbf{s}_{t-1} \odot \mathbf{a}_t \end{aligned} \tag{4.23}$$

where $\mathbf{W}_{ax}$, $\mathbf{W}_{ah}$ are weight matrices to be learned. $\mathbf{a}_t$ is called an $Adjustment$ gate since its task is to control what information of the given DA have been generated and what information should be retained for future time steps. Second, we consider how much the information preserved in the DA $\mathbf{s}_t$ can be contributed to the output, in which an additional output is computed by applying the output gate $\mathbf{o}_t$ on the remaining information in $\mathbf{s}_t$ as follows:

$$\begin{aligned} \mathbf{c}_a &= \mathbf{W}_{os}\mathbf{s}_t \\ \tilde{\mathbf{h}}_a &= \mathbf{o}_t \odot \tanh(\mathbf{c}_a) \end{aligned} \tag{4.24}$$

where $\mathbf{W}_{os}$ is a weight matrix to project the DA presentation into the output space, $\tilde{\mathbf{h}}_a$ is the Adjustment cell output. Final RALSTM output is a combination of both outputs of the traditional LSTM cell and the Adjustment cell, and computed as follows:

$$\mathbf{h}_t = \tilde{\mathbf{h}}_t + \tilde{\mathbf{h}}_a \tag{4.25}$$

Finally, the output distribution is computed by applying a softmax function g, and the distribution can be sampled to obtain the next token,

$$\begin{aligned} P(w_{t+1} \mid w_t, ...w_0, \mathbf{DA}) &= g(\mathbf{W}_{ho}\mathbf{h}_t) \\ w_{t+1} &\sim P(w_{t+1} \mid w_t, w_{t-1}, ...w_0, \mathbf{DA}) \end{aligned} \tag{4.26}$$

where $\mathbf{DA} = (\mathbf{s}, \mathbf{z})$.

4.4 Experiments

We extensively conducted a set of experiments to assess the effectiveness of the proposed models by using several metrics, datasets, and model architectures, in order to compare to prior methods.

4.4.1 Experimental Setups

The generators were implemented using the TensorFlow library (Abadi et al., 2016). The training and decoding procedures are described in Sections 2.5.1 and 2.5.2, respectively. The hidden layer size was set to be 80, and the generators were trained with a 70% of keep dropout rate. In order to better understand the effectiveness of our proposed methods, we: (i) performed an ablation experiments to demonstrate the contribution of each proposed components (Tables 4.1,

4.2), (ii) trained the models on the unseen Laptop, TV domains with varied proportion of training data, starting from 10% to 100% (Figure 4.6, 4.7), (iii) trained general models by merging all the data from four domains together and tested them in each individual domain (Figure 4.10), (iv) trained adaptation models on the union dataset of Restaurant and Hotel domains, then fine tuned the model on Laptop domain with varied amount of adaptation data (Figure 4.11), and (v) trained the generators on the unseen Laptop domain from scratch (*Scratch*), trained adaptation models by pooling out-of-domain Restaurant and Hotel (*Adapt-RH*), and pooling all three datasets Restaurant, Hotel and TV together (Figure 4.12, 4.13).

4.4.2 Evaluation Metrics and Baselines

The generator performance was assessed on the two evaluation metrics: the BLEU and the slot error rate ERR by adopting code from an open source benchmark toolkit for Natural Language Generation[4]. We compared the proposed models against three strong baselines which have been recently published as state-of-the-art NLG benchmarks[4].

- HLSTM proposed by (Wen et al., 2015a) which used a heuristic gate to ensure that all of the slot-value information was accurately captured when generating.
- SCLSTM proposed by (Wen et al., 2015b) which can jointly learn the gating signal and language model.
- RAOGRU proposed by (Tran and Nguyen, 2018d) which introduced a variety of gates to select, control the semantic elements, and generate the required sentence.
- ENCDEC proposed by (Wen et al., 2016b) which applied the attention-based encoder-decoder architecture.

4.5 Results and Analysis

We conducted extensive experiments on our proposed models and compared against the previous methods. Overall, the proposed models consistently achieve the better performance regarding both evaluation metrics (BLEU and ERR) across all domains in all test cases.

4.5.1 The Overall Model Comparison

Table 4.1 shows a comparison between the ARED-based models (denoted by $^\sharp$) in which the proposed models (in Row 2, 3) not only have better performance with higher the BLEU score but also significantly reduce the slot error rate ERR score by a large margin about 2% to 4% in every dataset.

The ARoA-M model shows the best performance among the EAD variants (in Table 4.1-Row 2) over all the four domains, while it is an interesting observation that the GR-ADD model with a simple addition operator for Refiner obtains the second best performance. These above prove the importance of the proposed component Refiner in aggregating and selecting the semantic elements.

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

Table 4.1: Performance comparison on four datasets in terms of the BLEU and the error rate ERR(%) scores. The results were produced by training each network on 5 random initialization and selected model with the highest validation BLEU score. ♯ denotes the Attention-based Encoder-Decoder model. The best and second best models highlighted in **bold** and ***italic*** face, respectively.

Model	**Restaurant**		**Hotel**		**Laptop**		**TV**	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
ENCDEC♯	0.7398	2.78%	0.8549	4.69%	0.5108	4.04%	0.5182	3.18%
HLSTM	0.7466	0.74%	0.8504	2.67%	0.5134	1.10%	0.5250	2.50%
SCLSTM	0.7525	0.38%	0.8482	3.07%	0.5116	0.79%	0.5265	2.31%
RAOGRU	***0.7762***	0.38%	0.8907	**0.17**%	0.5227	***0.47***%	0.5387	**0.60**%
GR-ADD♯	0.7742	0.59%	0.8848	1.54%	0.5221	0.54%	0.5348	0.77%
GR-MUL♯	0.7697	0.47%	0.8854	1.47%	0.5200	1.15%	0.5349	0.65%
ARoA-V♯	0.7667	0.32%	0.8814	0.97%	0.5195	0.56%	0.5369	0.81%
ARoA-M♯	0.7755	0.30%	0.8920	1.13%	0.5223	***0.50***%	***0.5394***	**0.60**%
ARoA-C♯	0.7745	0.45%	0.8878	1.31%	0.5201	0.88%	0.5351	***0.63***%
w/o A♯	0.7651	0.99%	0.8940	1.82%	0.5219	1.64%	0.5296	2.40%
w/o R♯	0.7748	***0.22***%	***0.8944***	0.48%	***0.5235***	0.57%	0.5350	0.72%
RALSTM♯	**0.7789**	**0.16**%	**0.8981**	***0.43***%	**0.5252**	**0.42**%	**0.5406**	***0.63***%

Row 1: Baselines, ***Row 2***: EAD variants, and ***Row 3***: RALSTM variants

Table 4.2: Performance comparison of the proposed models on four datasets in terms of the BLEU and the error rate ERR(%) scores. The results were averaged over 5 randomly initialized networks. The best and second best models highlighted in **bold** and ***italic*** face, respectively.

Model	**Restaurant**		**Hotel**		**Laptop**		**TV**	
	BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
ENCDEC♮	0.7358	2.98%	0.8537	4.78%	0.5101	4.24%	0.5142	3.38%
HLSTM	0.7436	0.85%	0.8488	2.79%	0.5130	1.15%	0.5240	2.65%
SCLSTM	0.7543	0.57%	0.8469	3.12%	0.5109	0.89%	0.5235	2.41%
RAOGRU	0.7730	0.49%	0.8903	**0.41**%	***0.5228***	***0.60***%	***0.5368***	0.72%
GR-ADD♮	0.7685	0.63%	0.8838	1.67%	0.5194	0.66%	0.5344	0.75%
GR-MUL§	0.7669	0.61%	0.8836	1.40%	0.5184	1.01%	0.5328	0.73%
ARoA-V♮	0.7673	0.62%	0.8817	1.27%	0.5185	0.73%	0.5336	0.68%
ARoA-M♮	0.7712	0.50%	0.8851	1.14%	0.5201	0.62%	0.5350	0.62%
ARoA-C♮	0.7690	0.70%	0.8835	1.44%	0.5181	0.78%	0.5307	0.64%
w/o A	0.7619	2.26%	***0.8913***	1.85%	0.5180	1.81%	0.5270	2.10%
w/o R	***0.7733***	***0.23***%	0.8901	0.59%	0.5208	***0.60***%	0.5321	***0.50***%
RALSTM	**0.7779**	**0.20**%	**0.8965**	***0.58***%	**0.5231**	**0.50**%	**0.5373**	**0.49**%

Row 1: Baselines, ***Row 2***: EAD variants, ***Row 3***: RALSTM variants

The ablation studies (Tables 4.1-Row 3, 4.2-Row 3) demonstrate the contribution of different model components in which the RALSTM models were assessed without Adjustment cell (*w/o A*), or without Refinement cell (*w/o R*). It clearly sees that the Adjustment cell contributes to reducing the slot error rate ERR score since it can effectively prevent the undesirable slot-value pair repetitions by gating the DA vector **s**. Moreover, a comparison between the models with gating the DA vector also indicates that the proposed models (*w/o R*, RALSTM) show significant improvements on both the evaluation metrics across the four domains com-

pared to the SCLSTM model. The RALSTM cell without the Refinement part is similar as the SCLSTM cell. However, it obtained the results much better than the SCLSTM baselines. This stipulates the necessary of the LSTM encoder and the Aligner in effectively partial learning the correlated order between slot-value representation in the DAs, especially for the unseen domain where there is only one training example for each DA. Table 4.2 further demonstrates the stable strength of our proposed models since the results' pattern stays unchanged compared to those in Table 4.1.

4.5.2 Model Comparison on an Unseen Domain

Figure 4.6, 4.7 show a comparison of five models (ENCDEC, SCLSTM, GR-ADD, ARoA-M, and RALSTM) which were trained from scratch on unseen Laptop (Figure 4.6) and TV (Figure 4.7) domains in a varied proportion of training data, start at 10% to 100%. It clearly shows that the BLEU increases while the slot error rate ERR decreases as more training data was fed. While the RALSTM outperforms the previous models in all cases, the ENCDEC has a much greater ERR score comparing to another models. Furthermore, our three proposed models produced much better the BLEU score in comparison with the two baselines, since their trend lines are always above the baselines with a large gap. All these prove the importance of the proposed components: the Refinement cell in aggregating and selecting the attentive information, and the Adjustment cell in controlling the feature vector (see examples in Figure 4.8).

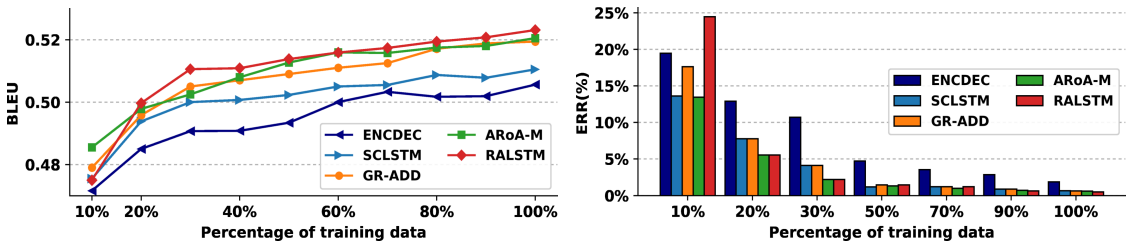


Figure 4.6: Performance comparison of the models trained on (unseen) Laptop domain.

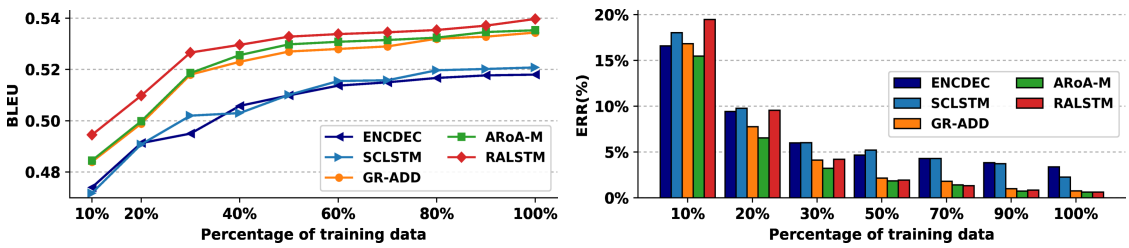
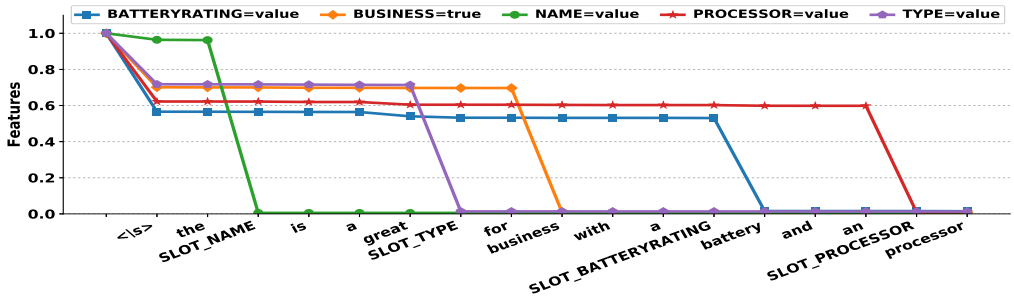


Figure 4.7: Performance comparison of the models trained on (unseen) TV domain.

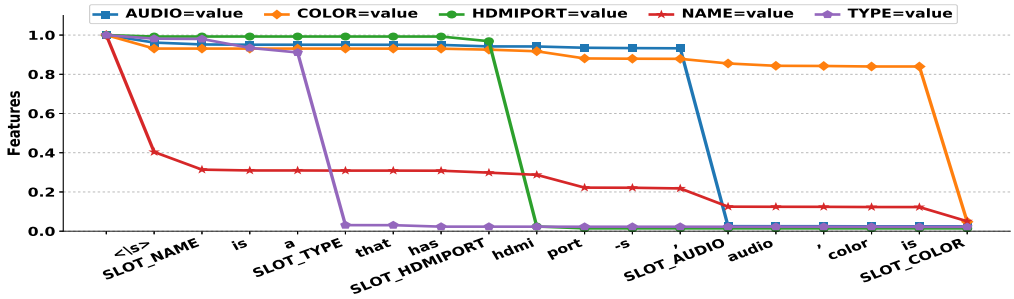
4.5.3 Controlling the Dialogue Act

One of the key features of the proposed models to substantial decreasing the slot error rate ERR score is their ability to efficiently control the Dialogue Act vector during generation. While the RALSTM model can control the DA started with a 1-hot vector representation via a gating

mechanism (Equation 4.20), the EAD-based models can attend to the DA semantic elements by utilizing an attention mechanism (Equations 4.9,4.10,4.11). On the one hand, Figure 4.8 shows an ability of the RALSTM model to effectively drive down the DA feature value vector **s** step-by-step, in which the model shows its ability to detect words and phrases describing a corresponding slot-value pair. On the other hand, Figure 4.9 illustrates a different attention behavior of ARoA-based models in the sentence, in which while all three ARoA-based models could capture the slot tokens and their surrounding words, the ARoA-C model with context shows its ability in attending the consecutive words.



(a) An example from the Laptop domain.



(b) An example from the TV domain.

Figure 4.8: Examples showing how RALSTM drives down the DA feature value vector **s** step-by-step, in which the model generally shows its ability to detect words and phases describing a corresponding slot-value pair.

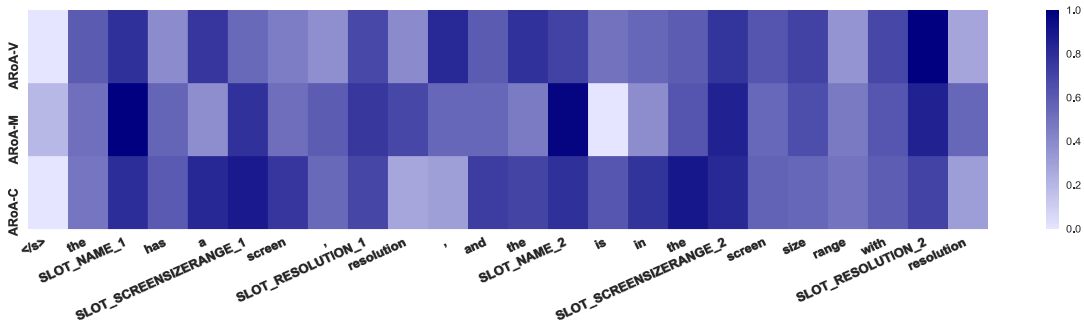


Figure 4.9: A comparison on attention behavior of three EAD-based models in a sentence on a given ***DA*** with sequence of slots [*Name_1, ScreenSizeRange_1, Resolution_1, Name_2, ScreenSizeRange_2, Resolution_2*].

4.5.4 General Models

Figure 4.10 shows a comparison performance of general models as described in Section 4.4.1. The results are consistent with the Figure 4.6, in which the proposed models (RALSTM, GR-ADD, and ARoA-M) have better performance than the ENCDEC and SCLSTM models on all domains in terms of the BLEU and the ERR scores, while the ENCDEC has difficulty in reducing the slot error rate. This indicates the relevant contribution of the proposed component Refinement and Adjustment cells to the original ARED architecture, in which the Refinement/Refiner with attention mechanism can efficiently select and aggregate the information before putting them into the traditional LSTM/GRU cell, while the Adjustment with gating DA vector can effectively control the information flow during generation.

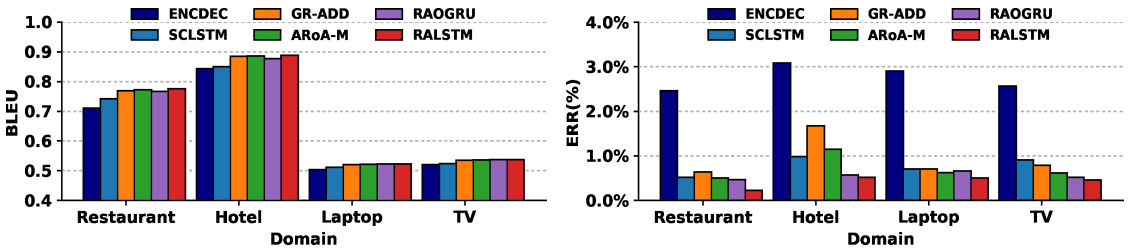


Figure 4.10: Performance comparison of the general models on four different domains.

4.5.5 Adaptation Models

In this experiments, we examine the ability of the generators in domain adaptation. Firstly, we trained the five generators with out-of-domain data by pooling the Restaurant and Hotel datasets together. We then varied the amount of in-domain training Laptop dataset and fine-tuned the model parameters. The results are presented in Fig. 4.11. The proposed models (GR-ADD, ARoR-M, and RALSTM) again outperform the baselines (SCLSTM, ENCDEC) irrespective of the size of the in-domain training data, especially the RALSTM model outperforms the other models in both cases where the sufficient in-domain data is used (as in Figure 4.11-*left*), and the limited in-domain data is available (Figure 4.11-*right*). These signal that the proposed models can adapt to a new, unseen domain faster than the previous ones.

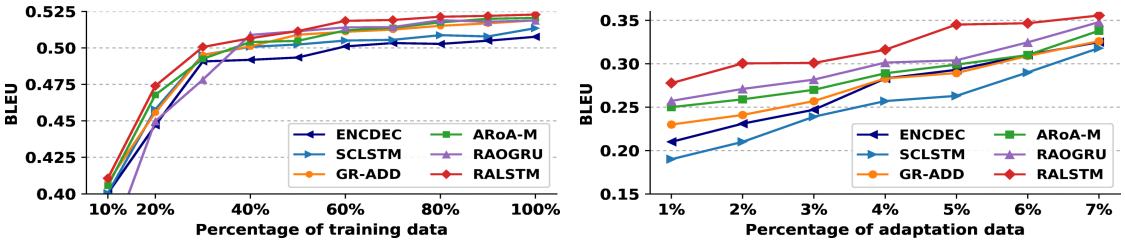


Figure 4.11: Performance on Laptop domain with varied amount of the adaptation training data when adapting models trained on union of Restaurant+Hotel datasets.

Secondly, we tested whether the proposed generators (ARoA-M and RALSTM) can leverage out-of-domain data on scenario of domain scalability. We trained the generator on the

unseen Laptop domain from scratch (*Scratch*), or trained adaptation models by pooling out-of-domain Restaurant and Hotel (*Adapt-RH*), or pooling all three datasets Restaurant, Hotel and TV together (*Adapt-RHT*) (Figures 4.12, 4.13). Both ARoA-M and RALSTM show its ability to leverage the existing resources since the adaptation models have better performance than the model trained from scratch. Figure 4.12 illustrates that when only a limited amount of in-domain data was available (<8%), the *Adapt-RH* pretrained with out-of-domain datasets achieve better results than the *Scratch* model trained with only in-domain data. Especially, adding more the TV dataset, which is similar to Laptop domain, to train adaptation model can benefit the adaptation model since the *Adapt-RHT* can learn faster than the *Adapt-RH* (Fig. 4.12-*right*). Figure 4.13 also demontrates the same trend with those in Figure 4.12 for ARoA-M model. Exceptionally, it is a bit surprising that when increasing the data of in-domain (<30%), the *Scratch* outperformed the two others. However, when a sufficient proportion of in-domain is used, (>30%), the *Adapt-RHT* consistently outperformed the *Scratch* model.

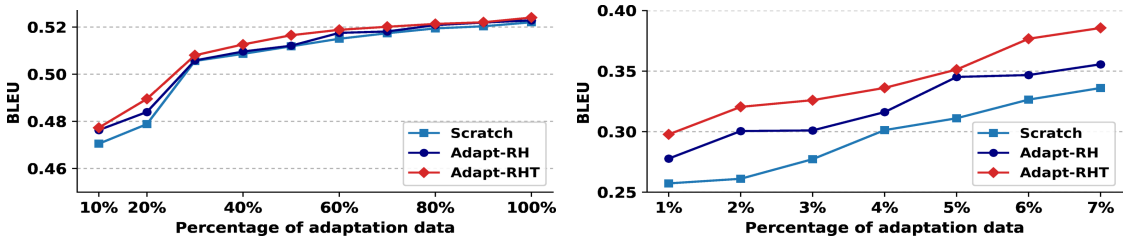


Figure 4.12: Performance evaluated on Laptop domain. Comparing ***RASLTM*** model trained from scratch (*Scratch*) with adaptation model trained on Restaurant+Hotel (*Adapt-RH*) and Restaurant+Hotel+Tv (*Adapt-RHT*) datasets.

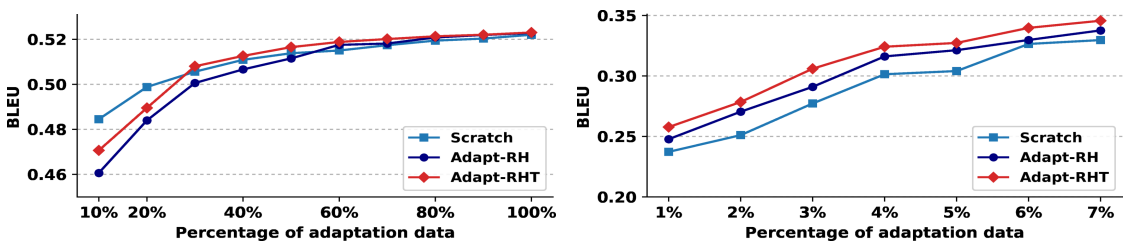


Figure 4.13: Performance evaluated on Laptop domain. Comparing ***ARoA-M*** model trained from scratch (*Scratch*) with adaptation model trained on Restaurant+Hotel (*Adapt-RH*) and Restaurant+Hotel+Tv (*Adapt-RHT*) datasets.

All these demonstrate the effectiveness in applying another attention over attentive input elements in the EAD-based models, and also the effectiveness in putting an Adjustment cell over the LSTM cell to control the DA vector. This additional components bring benefits to the ARED-based models of not only faster generalizing to a new domain but also effectively leveraging the out-of-domain data in order to extend to an open domain when only a very limited amount of in-domain data is available.

4.5.6 Model Comparison on Generated Utterances

A comparison of top responses generated for some input DAs between different models are shown in Tables 4.3, 4.4. While the previous models (HLSTM, ENCDEC, SCLSTM) still

produce some errors (missing and misplaced information), the proposed models (RALSTM and the models *All2** trained by pooling all datasets together) can generate appropriate sentences.

The one-layer attention-based models (ENCDEC, GR-MUL) likely tend to create misplaced slot-value for the comparing dialogue act style. However, by stacking another attention layer over the existing one (RAoA-M, ARoA-C) or controlling the 1-hot dialogue act representation vector (RALSTM, All2*) the proposed models seem to correct the slot-value generation order (Table 4.3). The *second attention* layer and the *adjustment* cell provided assistance in attending to the slot-value pair in place and preventing the missing information during generation. Table 4.4 further confirms that by combining the gating and attention mechanisms, and controlling the DA vector the proposed models can also generate more complete and concise responses than the previous methods.

Table 4.3: Comparison of top responses generated for some input dialogue acts between different models. Errors are marked in colors ([missing], misplaced information). [OK] denotes successful generation. *All2Laptop* is general model.

Model	**Generated Responses from Laptop Domain**
Input DA	?compare(name="*satellite notus 19*", pricerange="*budget*", drive="500 gb", name="portege thanatos 98", pricerange="expensive", drive="128 gb")
Reference	the *satellite notus 19* has a *500 gb* drive and is in the *budget* price range . on the other hand the *portege thanatos 98* has a *128 gb* drive and is in the *expensive* price range . which would you prefer
HLSTM	the *satellite notus 19* is a *budget* priced laptop with a *500 gb* drive and a 128 gb drive . which one do you prefer [portege thanatos 98][expensive]
ENCDEC	the *satellite notus 19* is in the *budget* price range . the *portege thanatos 98* has a 500 gb drive and is in the *expensive* price range . which one do you prefer [128 gb]
SCLSTM	the *satellite notus 19* is in the *budget* price range and has a *500 gb* drive and the *portege thanatos 98* has a *128 gb* drive . which one do you prefer [expensive]
GR-ADD	the *satellite notus 19* is in the *budget* price range with a *500 gb* drive , the *portege thanatos 98* has a *128 gb* drive and is in the *expensive* price range . which one do you prefer [OK]
GR-MUL	the *satellite notus 19* has a *500 gb* drive and is in the *budget* price range with a 128 gb drive and is in the *expensive* price range . which one do you prefer [portege thanatos 98]
ARoA-V	the *satellite notus 19* is in the *budget* price range and has a *500 gb* drive and is in the expensive price range . the *portege thanatos 98* has a *128 gb* drive
ARoA-M	the *satellite notus 19* is in the *budget* price range and has a *500 gb* drive , the *portege thanatos 98* has a *128 gb* drive and is in the *expensive* price range. which one do you prefer [OK]
ARoA-C	the *satellite notus 19* has a *500 gb* drive and is in the *budget* price range , the *portege thanatos 98* has a *128 gb* drive and is in the *expensive* price range . which one do you prefer [OK]
RALSTM	the *satellite notus 19* is in the *budget* price range with a *500 gb* drive , while the *portege thanatos 98* is in the *expensive* price range and has a *128 gb* drive . which one do you prefer [OK]
All2Laptop	the *satellite notus 19* is a *budget* price range with a *500 gb* drive and the *portege thanatos 98* has a *128 gb* drive and is in the *expensive* price range . which one do you prefer [OK]

4.6 Conclusion

This chapter 4 has presented a unified hybrid neural language generation framework which is an extension of ARED architecture, from which we introduce two novel generators to tackle the NLG problems of adequacy, completeness and adaptability. In the first proposed model, named Encoder-Aggregator-Decoder, a Refiner cell is introduced to align and control the semantic elements produced by an attention mechanism over the input elements. We also present several different choices of gating and attention mechanism which can be effectively applied to the Refiner. In the second model, a RALSTM component is introduced to select and aggregate semantic elements produced by the encoder and to generate the required utterances.

Table 4.4: Comparison of top responses generated for some input dialogue acts between different models. Errors are marked in color such as [missing] information. [OK] denotes successful generation. *All2Tv* is a general model.

Model	**Generated Responses from TV Domain**
Input DA	inform_count(count="*73*", type="*television*", hasusbport="*dontcare*", hdmiport="2", screensizerange="*dontcare*")
Reference	there are *73 televisions* if you *do not care* about whether it has *usb* port or not, with *2 hdmi ports* and if you *do not care* about the *screen size*
HLSTM	there are *73 televisions* with *2 hdmi ports* and if you *do not care* about the number of *usb* ports [screensizerange]
ENCDEC	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *usb* ports [screensizerange]
SCLSTM	there are *73 televisions* with *2 hdmi ports* if you *do not care* about the *screen size* and if you *do not care* about the number of *usb* ports [OK]
GR-ADD	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *screen size* or *usb* ports [OK]
GR-MUL	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *screen size* or *usb* ports [OK]
ARoA-V	there are *73 televisions* with *2 hdmi ports* , if you *do not care* about *screen size* or *usb* ports [OK]
ARoA-M	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *screen size* or *usb* ports [OK]
ARoA-C	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *screen size* or *usb* ports [OK]
RALSTM	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *screen size* or *usb* ports [OK]
All2Tv	there are *73 televisions* with *2 hdmi ports* if you *do not care* about *screen size* or *usb* ports [OK]

Both proposed generators can jointly train both sentence planning and surface realization to produce natural language sentences. We assessed the proposed models on four NLG domains (hotel, restaurant, tv, and laptop) and compared to strong previous generators. The proposed models empirically show consistent improvement over the previous methods in both the BLEU and ERR evaluation metrics in terms of adequacy and completeness. The experimental results also showed that the proposed models have an ability to adapt to a new, unseen domain with a limited amount of in-domain data and show advantage from out-of-domain data. Nevertheless, the *hybrid*-based models are not explicitly designed for work in the case of *low-resource* setting which can easily cause the performance degradation. In the next chapter, we discuss an effective way to construct NLG systems that can work acceptably well with only a modest amount of data.

Chapter 5

Variational Model for Low-Resource NLG

In this chapter, we present approaches dealing with the problem of *low-resource* setting data. Despite the fact that previous models have shown to work well when providing a sufficient in-domain data, *low-resource* data can easily harm the generators' performance (see Table 5.1). Furthermore, prior NLG systems often require a well-defined ontology dataset which is highly expensive and time-consuming to collect. Thus, there is a need to develop NLG systems that can work acceptably well on a small training dataset.

Table 5.1: Results evaluated on four domains by training models from *scratch* with *100%* and *10%* amount of in-domain data, respectively. The models work well when providing a sufficient in-domain data (*sec.* **scr100**), while *low-resource* setting data really harms the models performance (*sec.* **scr10**).

	Model	**Hotel**		**Restaurant**		**Tv**		**Laptop**	
		BLEU	ERR	BLEU	ERR	BLEU	ERR	BLEU	ERR
scr100	HLSTM	0.8488	2.79%	0.7436	0.85%	0.5240	2.65%	0.5130	1.15%
	SCLSTM	0.8469	3.12%	0.7543	0.57%	0.5235	2.41%	0.5109	0.89%
	ENCDEC	0.8537	4.78%	0.7358	2.98%	0.5142	3.38%	0.5101	4.24%
	RALSTM	**0.8965**	**0.58%**	**0.7779**	**0.20%**	**0.5373**	**0.49%**	**0.5231**	**0.50%**
scr10	HLSTM	0.7483	8.69%	0.6586	6.93%	0.4819	9.39%	0.4813	7.37%
	SCLSTM	0.7626	17.42%	0.6446	16.93%	0.4290	31.87%	0.4729	15.89%
	ENCDEC	0.7370	23.19%	0.6174	23.63%	0.4570	21.28%	0.4604	29.86%
	RALSTM	0.6855	22.53%	0.6003	17.65%	0.4009	22.37%	0.4475	24.47%

The chapter demonstrates two potential solutions for above-mentioned problems, which are *domain adaptation* and *model designing for low-resource* training data.

Firstly, *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 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 (see Table 5.1, *sec.* **scr100**), while the target domain is assumed to have different characteristics 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 (see Table 5.1, *sec.* **scr10**). Hence, we mainly aim at achieving good performance on the target domain by leveraging the source data or adapting

model trained on the source domain. We publish this work in (Tran and Nguyen, 2018a).

Secondly, model designing for *low-resource setting* has not been studied well in the NLG literature. The generation models have achieved great performances irrespective of providing sufficient labeled datasets (Wen et al., 2015b,a; Tran et al., 2017a; Tran and Nguyen, 2017a). For *low-resource* scenario, one can further 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). However, we present in this chapter an explicit way to build a generator that can work acceptably well in case of having a modest amount of training data. This model is based on our work in (Tran and Nguyen, 2018b,e).

In summary, this chapter presents two approaches dealing with the problem of low-resource setting data. We first propose an adversarial training procedure to train multi-domain, variational generator via multiple adaptation steps, which enable the generator to learn more efficiently when in-domain data is in short supply. We then propose a combination of two VAEs, which enables the variational-based generator to learn more efficiently in low-resource setting data. In this chapter, 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 to guide the generator to generate outputs that resemble the sentences drawn from the target domain. The two critics 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 variational model that benefits the generator to not only achieve state-of-the-art over 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 training, scratch training, and unsupervised training with different amount of training dataset.

This chapter is organized as follows. Section 5.1 describes in detail a Variational Neural Language Generator framework. Section 5.2 presents an Adversarial Variational NLG (VDANLG) for domain adaptation, whereas Section 5.3 presents a dual variational model for low-resource setting in-domain data. The experiments are described in Section 5.4, whereas Section 5.5 shows results and analyses. We present our summary and discussion in Section 5.6.

5.1 VNLG - Variational Neural Language Generator

5.1.1 Variational Autoencoder

Variational autoencoder (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 $\mathbf{x}$ and the joint distribution is formulated as follows:

$$p(\mathbf{x}, z) = p_\theta(\mathbf{x}|z)p(z) \tag{5.1}$$

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-liner 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:

$$q_\phi(z|\mathbf{x}) = \mathcal{N}(\mu(x), \sigma^2(\mathbf{x})) \tag{5.2}$$

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

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

$$z = \mu + \sigma \odot \epsilon \tag{5.3}$$

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:

$$\mathcal{L}_{VAE}(\theta, \phi, \mathbf{x}) = -KL(q_\phi(z|\mathbf{x})||p_\theta(z)) + \mathbb{E}_{q_\phi(z|\mathbf{x})}[\log p_\theta(\mathbf{x}|z)] \leq \log p(\mathbf{x}) \tag{5.4}$$

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.

5.1.2 Variational Neural 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:

$$p(\mathbf{u}|\mathbf{d}) = \int_z p(\mathbf{u}, z|\mathbf{d})\mathbf{d}_z = \int_z p(\mathbf{u}|z, \mathbf{d})p(z|\mathbf{d})\mathbf{d}_z \tag{5.5}$$

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

$$\mathcal{L}_{\text{VAE}}(\theta, \phi, \mathbf{d}, \mathbf{u}) = -KL(q_\phi(z|\mathbf{d}, \mathbf{u})||p_\theta(z|\mathbf{d})) + \mathbb{E}_{q_\phi(z|\mathbf{d},\mathbf{u})}[\log p_\theta(\mathbf{u}|z, \mathbf{d})] \tag{5.6}$$

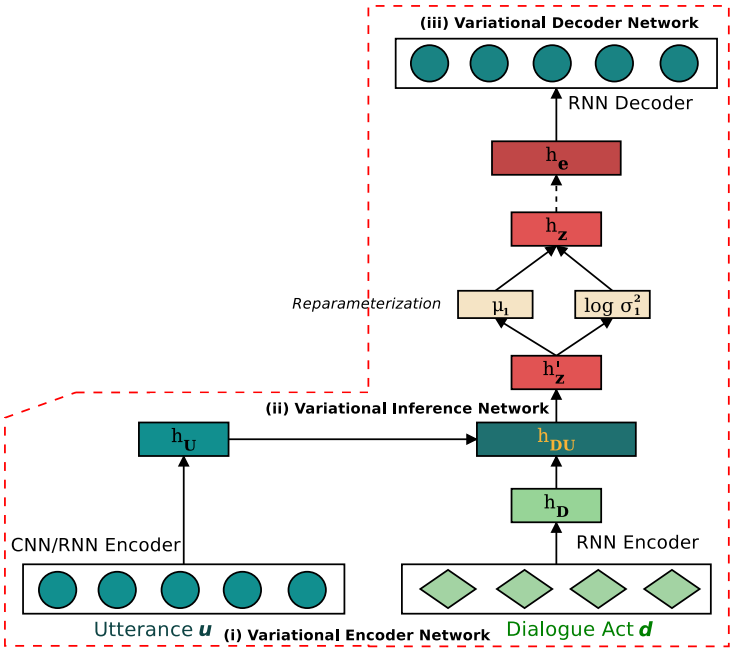


Figure 5.1: The Variational NLG architecture. The model consists of three main components: (i) Variational Encoder Network, (ii) Variational Inference Network, and (iii) Variational Decoder Network.

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.

The variational architecture for natural language generation is demonstrated in Figure 5.1, in which a variational inference is integrated into an encoder-decoder based natural language generator (see Section 4.1).

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 5.3. The CNN consists of *L*-1 convolutional layers and a *L*-th fully-connected layer, which aims at encoding an input utterance **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, $\mathbf{v}^{(j,l)} = \text{ReLU}(\mathbf{U} * \mathbf{W}_c^{(j,l)} + b^{(j,l)}) \in \mathbb{R}^{(T^{(l)}-h)/s^{(l)}+1}$, 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 $\mathbf{V}^{(l)} = [v^{(1,l)}, ..., v^{(k_l,l)}] \in \mathbb{R}^{k_l \times [(T^{(l)}-h)/s^{(l)}+1]}$. 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 substructure 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 $\mathbf{h_U}$.

RNN Utterance Encoder

For constructing a domain-adaptation generator, we utilize RNN Utterance Encoder whose architecture is same as BiLSTM DA encoder in Subsection 5.1.2. 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 $\mathbf{h}'_i = \overrightarrow{\mathbf{h}'_i} + \overleftarrow{\mathbf{h}'_i}$

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 work we employ neural network to approximate the posterior distribution of z to simplify the posterior inference. We assume the approximation has the following form:

$$q_\phi(z|\mathbf{d}, \mathbf{u}) = \mathcal{N}(z; \mu_1(f(\mathbf{h}_D, \mathbf{h}_U)), \sigma_1^2(f(\mathbf{h}_D, \mathbf{h}_U))\boldsymbol{I}) \tag{5.7}$$

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:

$$\mathbf{h}'_z = f(\mathbf{h}_D, \mathbf{h}_U) = g(\mathbf{W}_z[\mathbf{h}_D; \mathbf{h}_U] + b_z) \tag{5.8}$$

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:

$$\mu_1 = \mathbf{W}_{\mu_1}\mathbf{h}'_z + b_{\mu_1}, \log \sigma_1^2 = \mathbf{W}_{\sigma_1}\mathbf{h}'_z + b_{\sigma_1} \tag{5.9}$$

where μ_1, $\log \sigma_1^2$ are both d_z dimension vectors.

Neural Prior Model

We model the prior as follows:

$$p_\theta(z|\mathbf{d}) = \mathcal{N}(z; \mu'_1(\mathbf{d}), \sigma'_1(\mathbf{d})^2\boldsymbol{I}) \tag{5.10}$$

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. 5.7 and Eq. 5.9, 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:

$$\mathbf{h}_z = \mu_1 + \sigma_1 \odot \epsilon, \epsilon \sim \mathcal{N}(0, \boldsymbol{I}) \tag{5.11}$$

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:

$$\mathbf{h}_e = g(\mathbf{W}_e\mathbf{h}_z + b_e) \tag{5.12}$$

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 5.3 and Figure 5.4).

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:

$$p(\mathbf{u}|z, \mathbf{d}) = \prod_{j=1}^{T_U} p(\mathbf{u}_t|\mathbf{u}_{<t}, z, \mathbf{d}) \tag{5.13}$$

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 study, 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 Figure 5.1-(iii). We modify the cell calculation as follows:

$$\begin{pmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \hat{\mathbf{c}}_t \end{pmatrix} = \begin{pmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{pmatrix} \mathbf{W}_{4d_h,4d_h} \begin{pmatrix} \mathbf{h}_e \\ \mathbf{d}_t \\ \mathbf{h}_{t-1} \\ \mathbf{u}_t \end{pmatrix} \tag{5.14}$$

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 Figure 5.1-(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 5.5.2 for details).

5.2 VDANLG - An Adversarial Domain Adaptation VNLG

In this Section, inspired by work of Chen et al. (2017) 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 Subsection 5.1.2) in this setting, resulting in a Variational Domain Adaptation NLG (VDANLG) model is illustrated in Figure 5.2.

5.2.1 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.

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.

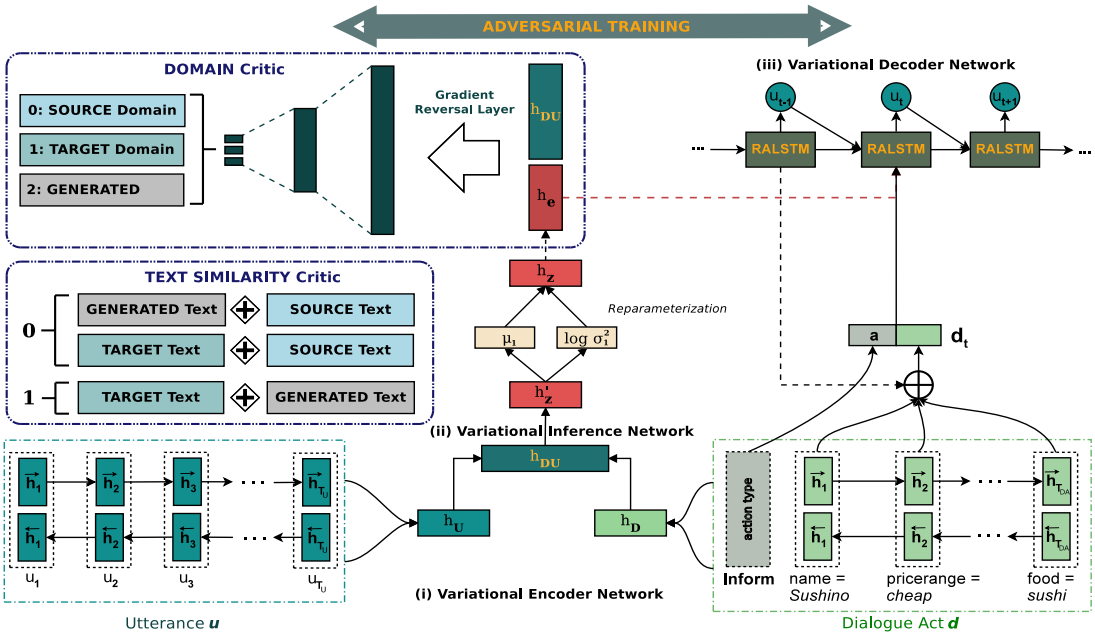


Figure 5.2: The Variational Domain Adaptation NLG (VDANLG) architecture. The model consists of two main components: the VIR-RALSTM to generate the sentence, which comprises : (i) Variational Encoder Network, (ii) Variational Infererence Network, and (iii) Variational Decoder Network; and two Critics with an adversarial training procedure to guide the model in domain adaptation, which is composed of a Domain critic and a Text Similarity critic.

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.

5.2.2 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 $\Theta_{\mathcal{T}}$ that can perform acceptably well on the target domain.

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:

$$\begin{aligned}
&\mathcal{L}_s(\psi) = -\sum_{n=1}^{N} \log C_s(l_s^n|\mathbf{u}_{(1)}^n, \mathbf{u}_{(2)}^n, \psi), \\
&l_s^n = \begin{cases} 1 - similar & \text{if } (\mathbf{u}_{(1)}^n, \mathbf{u}_{(2)}^n) \in \mathcal{P}_{sim}, \\ 0 - unsimilar & \text{if } (\mathbf{u}_{(1)}^n, \mathbf{u}_{(2)}^n) \in \mathcal{P}_{unsim}, \end{cases} \\
&\mathcal{U}_\mathcal{G} = \{\mathbf{u}|\mathbf{u} \sim \mathcal{G}(.|\mathbf{d}_\mathcal{T}, .)\}, \mathcal{P}_{sim} = \{\mathbf{u}_\mathcal{T}^n, \mathbf{u}_{\mathcal{U}_\mathcal{G}}^n\}, \mathcal{P}_{unsim} = (\{\mathbf{u}_\mathcal{T}^n, \mathbf{u}_\mathcal{S}^n\}, \{\mathbf{u}_{\mathcal{U}_\mathcal{G}}^n, \mathbf{u}_\mathcal{S}^n\})
\end{aligned} \tag{5.15}$$

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:

$$\mathcal{L}_d(\varphi) = -\sum_{n=1}^{N} \log C_d(l_d^n|\mathbf{d}^n, \mathbf{u}^n, \varphi), l_d^n = \begin{cases} source & \text{if } (\mathbf{d}^n, \mathbf{u}^n) \in (\mathcal{D}_\mathcal{S}, \mathcal{U}_\mathcal{S}), \\ target & \text{if } (\mathbf{d}^n, \mathbf{u}^n) \in (\mathcal{D}_\mathcal{T}, \mathcal{U}_\mathcal{T}), \\ generated & \text{if } (\mathbf{d}^n, \mathbf{u}^n) \in (\mathcal{D}_\mathcal{T}, \mathcal{U}_\mathcal{G}), \end{cases} \tag{5.16}$$

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.

Training Variational Neural Language Generator

We utilize the Monte Carlo method to approximate the expectation over the posterior in Eq. 5.6, *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:

$$\mathcal{L}(\theta, \phi, \mathbf{d}, \mathbf{u}) \simeq -KL(q_\phi(z|\mathbf{d}, \mathbf{u})||p_\theta(z|\mathbf{d})) + \frac{1}{M}\sum_{m=1}^{M}\sum_{t=1}^{T_u} \log p_\theta(\mathbf{u}_t|\mathbf{u}_{<t}, \mathbf{d}, \mathbf{h}_z^{(m)}) \tag{5.17}$$

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

Adversarial Training

Our domain adaptation architecture is demonstrated in Figure 5.1, in which both generator $\mathcal{G}$ and critics C_s, and C_d jointly train by pursuing competing goals as follows. Given a dialogue

act $\mathbf{d}_\mathcal{T}$ in the target domain, the generator generates K sentences $\mathbf{u}$'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.

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 $(\mathcal{D}_\mathcal{T}, \mathcal{U}_\mathcal{T})$;
1 Pretrain $\mathcal{G}$ on $(\mathcal{D}_\mathcal{S}, \mathcal{U}_\mathcal{S})$ using VIR-RALSTM (see Subsection 5.1.2);
2 **while** Θ *has not converged* **do**
3 **for** *i = 0, .., $N_\mathcal{T}$* **do**
4 Sample $(\mathbf{d}_\mathcal{S}, \mathbf{u}_\mathcal{S})$ from source domain;
5 (D_1)-Compute $g_d = \nabla_\varphi \mathcal{L}_d(\varphi)$ using Eq. 5.16 for $(\mathbf{d}_\mathcal{S}, \mathbf{u}_\mathcal{S})$ and $(\mathbf{d}_\mathcal{T}, \mathbf{u}_\mathcal{T})$;
6 (D_2)-Adam update of φ for C_d using g_d;
7 (G_1)-Compute $g_\mathcal{G} = \{\nabla_\theta \mathcal{L}(\theta, \phi), \nabla_\phi \mathcal{L}(\theta, \phi)\}$ using Eq. 5.17
8 (G_2)-Adam update of θ, ϕ for $\mathcal{G}$ using $g_\mathcal{G}$
9 (S_1)-Compute $g_s = \nabla_\psi \mathcal{L}_s(\psi)$ using Eq. 5.15 for $(\mathbf{u}_\mathcal{T}, \mathbf{u}_\mathcal{S})$;
10 (S_2)-Adam update of ψ for C_s using g_s;
11 $\mathcal{U}_\mathcal{G} \leftarrow \{\mathbf{u}_{\bar{k}}\}_{\bar{k}=1}^{K}$, where $\mathbf{u}_{\bar{k}} \sim \mathcal{G}(.|\mathbf{d}_\mathcal{T}^{(i)}, .)$;
12 Choose top $\boldsymbol{k}$ best sentences of $\mathcal{U}_\mathcal{G}$;
13 **for** *j = 1,..,**k*** **do**
14 (D_1), (D_2) steps for C_d with $(\mathbf{d}_\mathcal{T}, \mathbf{u}_\mathcal{G}^{(j)})$;
15 (S_1), (S_2) steps for C_s with $(\mathbf{u}_\mathcal{G}^{(j)}, \mathbf{u}_\mathcal{S})$ and $(\mathbf{u}_\mathcal{T}, \mathbf{u}_\mathcal{G}^{(j)})$;
16 **end**
17 **end**
18 **end**

Generally, the current generator $\mathcal{G}$ for each training iteration i takes a *target* dialogue act $\mathbf{d}_\mathcal{T}^{(i)}$ as input to over-generate a set $\mathcal{U}_\mathcal{G}$ 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 "*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 5.3).

5.3 DualVAE - A Dual Variational Model for Low-Resource Data

Starting from a Varitional neural language generator with a CNN utterance encoder (VIC-RALSTM) in Subsection 5.1.2, 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. The following Subsection 5.3.1 presents a second VAE model which is a Variationl CNN-DCNN model as shown in the left side of

Figure 5.3. Subsection 5.3.2 then proposes a novel training procedure to effectively leverage knowledge from a small amount of training data.

5.3.1 Variational CNN-DCNN Model

This VAE model (left side in Figure 5.3) 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 $\mathbf{h_U}$, we apply another linear regression to obtain the distribution parameter μ_2 and $\log \sigma_2^2$ as follows:

$$\mu_2 = \mathbf{W}_{\mu_2}\mathbf{h_U} + b_{\mu_2}, \log \sigma_2^2 = \mathbf{W}_{\sigma_2}\mathbf{h_U} + b_{\sigma_2} \tag{5.18}$$

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:

$$\mathbf{h}_{zu} = \mu_2 + \sigma_2 \odot \epsilon, \epsilon \sim \mathcal{N}(0, \boldsymbol{I}) \tag{5.19}$$

In order to integrate the latent variable $\mathbf{h}_{zu}$ into the DCNN Decoder, we use a shared non-linear transformation as in Eq. 5.12 (dashed black line in Figure 5.3):

$$\mathbf{h}_e = g(\mathbf{W}_e\mathbf{h}_{zu} + b_e) \tag{5.20}$$

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:

$$p(\hat{\mathbf{u}}_t = s) = \frac{exp\{\tau^{-1}cos(\hat{\mathbf{u}}_t, \mathbf{E}[s])\}}{\sum_{s' \in \mathcal{V}} exp\{\tau^{-1}cos(\hat{\mathbf{u}}_t, \mathbf{E}[s'])\}} \tag{5.21}$$

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.

The resulting model named DualVAE by incorporating the Variational NLG (VIC-RALSTM) with the Variational CNN-DCNN model and depicted in Figure 5.3.

5.3.2 Training Dual Latent Variable Model

Training Variational Language Generator

Similar to Subsection 5.2.2 on training a variational NLG, the joint training objective for a training instance pair $(\mathbf{d}, \mathbf{u})$ is formulated as follows:

$$\mathcal{L}_{\text{VIC-RALSTM}} = \mathcal{L}(\theta, \phi, \mathbf{d}, \mathbf{u}) \tag{5.22}$$

where $\mathcal{L}(\theta, \phi, \mathbf{d}, \mathbf{u})$ is computed as in Eq. 5.17.

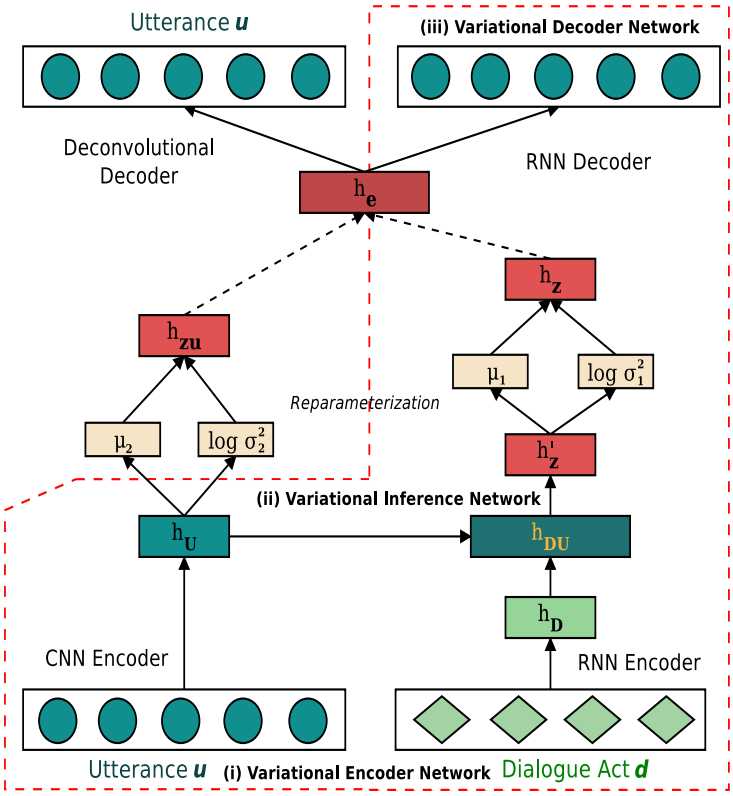


Figure 5.3: The Dual Variational Model consists of two VAE models: (I) the Variational Natural Language Generator (VIC-RALSTM) in the dashed red box to generate utterances, which comprises: (i) Variational Encoder Network, (ii) Variational Inference Network, and (iii) Variational Decoder Network; and (II) the Variational CNN-DCNN Model (left side) which is composed of a CNN Encoder and a Deconvolutional Decoder. The CNN Encoder for utterance encoding is shared between the two VAEs.

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:

$$\mathcal{L}_{\text{CNN-DCNN}} = \mathcal{L}(\theta', \phi', \mathbf{u}) = -KL(q_{\phi'}(z|\mathbf{u})||p_{\theta'}(z)) + \mathbb{E}_{q_{\phi'}(z|\mathbf{u})}[\log p_{\theta'}(\mathbf{u}|z)] \leq \log p(\mathbf{u}) \quad (5.23)$$

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.

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:

$$\mathcal{L}_{\text{DualVAE}} = \mathcal{L}_{\text{VIC-RALSTM}} + \alpha \mathcal{L}_{\text{CNN-DCNN}} \quad (5.24)$$

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.

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:

$$\mathcal{L}_{\text{VIC-DCNN}} = \mathcal{L}(\theta', \phi, \mathbf{d}, \mathbf{u}) \simeq -KL(q_\phi(z|\mathbf{d}, \mathbf{u})||p_{\theta'}(z|\mathbf{d})) + \mathbb{E}_{q_\phi(z|\mathbf{d},\mathbf{u})}[\log p_{\theta'}(\mathbf{u}|z, \mathbf{d})], \quad (5.25)$$

and a joint cross training objective is employed:

$$\mathcal{L}_{\text{CrossVAE}} = \mathcal{L}_{\text{VIC-RALSTM}} + \alpha(\mathcal{L}_{\text{CNN-DCNN}} + \mathcal{L}_{\text{VIC-DCNN}}) \quad (5.26)$$

5.4 Experiments

5.4.1 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 is initially set to be 0.001, and after 3 epochs for the Union dataset and 5 epochs for the single dataset the learning rate is decayed every epoch using an exponential rate of 0.95. For the variational inference, we set the latent variable size to be 16 for VDANLG model and 300 for dual VAEs.

5.4.2 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., 2015), 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.

5.4.3 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 5.2.1) 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:

$$\begin{aligned} p &= \text{float}(i)/num\_steps, \\ \lambda_p &= \frac{2}{1 + \exp(-10 * p)} - 1 \end{aligned} \quad (5.27)$$

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.

5.4.4 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[1]. We compared the proposed models against strong baselines which have been recently published as NLG benchmarks of the above datasets.

- Gating-based generators, including HLSTM (Wen et al., 2015a) and SCLSTM (Wen et al., 2015b).
- Attention-based generators, including ENCDEC (Wen et al., 2016b) and RALSTM (Tran and Nguyen, 2017a).

5.5 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 Subsection 5.5.1. Subsections 5.5.2 and 5.5.3 present the results and analyses of domain adaptation models and dual variational models, respectively.

5.5.1 Integrating Variational Inference

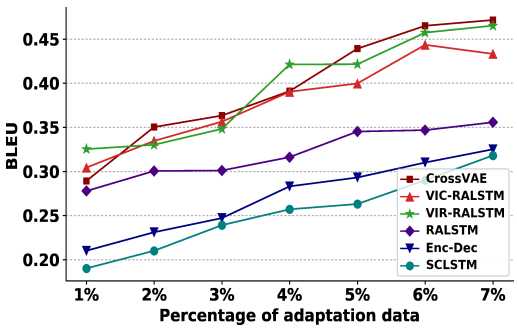


Figure 5.4: Performance on **Laptop** domain with varied limited amount, from 1% to 7%, of the adaptation training data when adapting models pre-trained on [Restaurant+Hotel] union dataset.

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

Table 5.2: 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%
ENCDEC (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)	0.8965	0.58%	0.7779	0.20%	0.5373	0.49%	0.5231	0.50%
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%

We compare the original model RALSTM with its modification by integrating with Variational Inference (VIR-RALSTM and VIC-RALSTM) as demonstrated in Figure 5.4 and Table 5.2. 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 5.2, *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% (Figure 5.4). Table 5.2, *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 5.5.3. The following Section 5.5.2 provides in detail results and analyses of the VDANLG model in tackling domain adaptation problems.

5.5.2 Adversarial VNLG for Domain Adaptation

We compared the Variational Domain Adaptation NLG (see Section 5.2) 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.

Table 5.3: Ablation studies' results evaluated on **Target** domains by *adaptation* training proposed models from Source domains using only *10%* amount of the **Target** domain data (*sec.* 1, 2, 4, 5). The results were averaged over 5 randomly initialized networks.

	Target / Source	Hotel		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%	-	-	-	-

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

Ablation Studies

The ablation studies (Table 5.3, *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 clearly sees that, in comparison to models trained without Critics in Table 5.3 *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 pairs. A comparison of model adapting from source Laptop domain between VIR-RALSTM without Critics (Laptop in *sec.* 1) and VDANLG (Laptop in *sec.* 2) 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 5.2), the performances are extremely impaired when training from *scratch* with only limited amount of in-domain training data.

Table 5.3 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.

Adaptation versus scr100 Training Scenario

It is interesting to compare *adaptation* (Table 5.3, *sec.* 2) with *scr100* training scenario (Table 5.2). The VDANLG model shows its considerable ability to shift to another domain with a limited of in-domain labels whose results are competitive to or in some cases better than the previous models trained on full labels of the **Target** domain. A specific comparison evaluated on the **Tv** domain where the VDANLG model trained on the source Laptop (*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 ENCDEC (51.42, 3.38). The VADNLG models, in many cases, also have lower of slot error rate ERR results than the ENCDEC model. These indicate the stable strength of the VDANLG models in adapting to a new domain when the target domain data is scarce.

Distance of Dataset Pairs

To better understand the effectiveness of the methods, we analyze the learning behavior of the proposed model between different dataset pairs. The datasets' order of difficulty was, from easiest to hardest: Hotel ↔ Restaurant↔ Tv ↔ Laptop. On the one hand, it might be said that the *longer* datasets' distance is, the more difficult of domain adaptation task becomes. This clearly shows in Table 5.3, *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 5.3, *sec.* 1, 2). The distance of datasets is also shown via the differences of word-level distribution using word clouds in Figure 2.2.

Table 5.3, *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 5.3, *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 5.3, *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.

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 5.4, despite the fact that the models were instead adaptation trained on the **Counterfeit** datasets, or in other words, were indirectly trained on the (*Test*) domains. However, the proposed models still showed positive signs in remarkably reducing the slot error rate ERR in the cases of *Hotel* and *Tv* be the (*Test*) domains. Surprisingly, even the source domains (Hotel/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).

Table 5.4: Results evaluated on (*Test*) domains by *Unsupervised* adapting VDANLG from Source domains using only **10%** of the **Target** domain **Counterfeit X2Y** where {**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%	-	-

Comparison on Generated Outputs

We present top responses generated for different scenarios from Laptop (Table 5.5) and TV (Table 5.6) 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 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 5.5, 5.6) 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

Table 5.5: 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$^{\sharp}$ = VIR-RALSTM$^{\flat}$+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$^{\flat}$	*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$^{\flat}$	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$^{\flat}$	*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]$^{\flat}$	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$^{\sharp}$	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$^{\sharp}$	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$^{\sharp}$	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]$^{\sharp}$	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
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$^{\flat}$	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$^{\flat}$	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$^{\flat}$	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]$^{\flat}$	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$^{\sharp}$	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$^{\sharp}$	*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$^{\sharp}$	*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]$^{\sharp}$	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]

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 5.5 contains all the

Table 5.6: 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$^{\sharp}$ = VIR-RALSTM$^{\flat}$+SC+DC.

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

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 5.5-Example 2 generated by the Hotel$^{\sharp}$, Tv$^{\sharp}$, [R+H]$^{\sharp}$ models, and a sample generated by the Laptop$^{\sharp}$ in Table 5.6 are all fulfilled responses. An analysis of generated responses in Table 5.5-Exmple 2 illustrates that the VDANLG models seem to generate a concise response since the models show a tendency to form some potential slots into a concise phrase, *i.e.* "SLOT_NAME SLOT_TYPE". For example, the VDANLG models tend to concisely response as "the *portege phosphorus 43 laptop* ..." instead of "the *portege phosphorus 43* is a *laptop* ...".

All these above demonstrate that the VDANLG models have ability to work acceptably well in the low-resource setting since they produce better results with a much lower score of the slot error rate ERR.

5.5.3 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.

Ablation Studies

The ablation studies (Table 5.7) 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 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.

Table 5.7: 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%
	ENCDEC (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)	**0.8965**	0.58%	***0.7779***	***0.20%***	***0.5373***	***0.49%***	***0.5231***	**0.50%**
	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)	***0.8926***	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%
	ENCDEC (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%
	ENCDEC (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%**

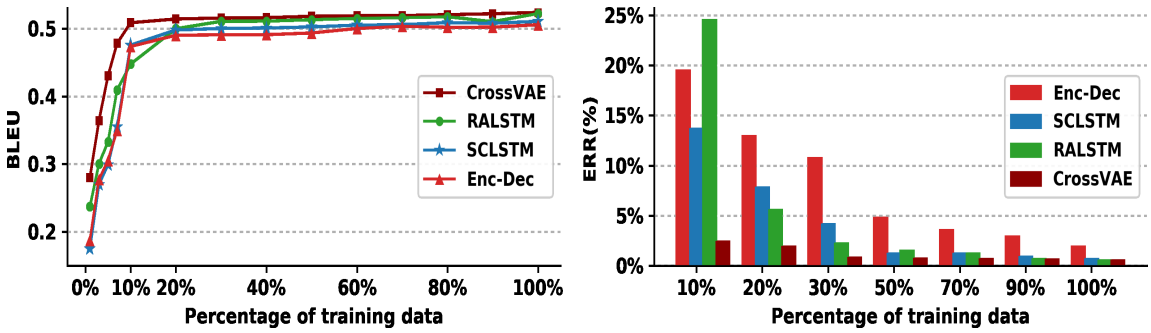
On the other hand, the previous methods have extremely impaired performances regarding low BLEU score and high slot error rate ERR when training the models from *scratch* with insufficient in-domain data (*scr10*). In contrast, by integrating the variational inference, the VIC-RALSTM model can significantly 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 ENCDEC 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 ENCDEC (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 ENCDEC (3.38 ERR), or DualVAE (2.44 ERR) and ENCDEC (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 ENCDEC (51.42 BLEU, 3.38 ERR).

Model comparison on unseen domain

Figure 5.5: Performance comparison of the models trained on Laptop domain.



In this experiment, we trained four models (ENCDEC, 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 Figure 5.5. 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 (ENCDEC, SCLSTM, RALSTM) in all cases. While the performance of the CrossVAE, RALSTM model starts to saturate around 30% and 50%, respectively, the ENCDEC 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.

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 5.8.

Table 5.8: Results evaluated on **Target** domains: by *adaptation* training SCLSTM model from 100% (denoted as $\flat$) of Source data, and the CrossVAE model from 30% (denoted as $\sharp$), 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%

Both SCLSMT and CrossVAE models can take advantage of "*close*" dataset pairs, *i.e.*, Restaurant $\leftrightarrow$ Hotel, and Tv $\leftrightarrow$ Laptop, to achieve better performances compared to those of the "*different*" dataset pairs, *i.e.* Latop $\leftrightarrow$ Restaurant. The SCLSTM (denoted by $\flat$) 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 $\sharp$) which is also better than the adaptation model trained on 100% out-of-domain data (denoted by ξ).

Preliminary experiments on semi-supervised training are 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 dialogue 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.

Comparison on Generated Outputs

We present top responses generated for different scenarios from TV (Table 5.9) and Laptop (Table 5.10) domains, which further demonstrate the effectiveness of the proposed methods.

Table 5.9: 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 ?
ENCDEC-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]
ENCDEC-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]
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*.
ENCDEC-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]
ENCDEC-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]

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 ENCDEC, HLSTM and SCLSTM models, for example, in Table 5.9 tend to generate outputs with redundant slots (*i.e.* *SLOT_HDMIPORT*, *SLOT_NAME*, SLOT_FAMILY), missing slots (*i.e.* [l7 fam-

ily], [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 5.9-Example 1, 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 5.9-Example 2, 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 for Laptop example in Table 5.10.

Table 5.10: 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 hour*'; drive='*500 gb*'; name='*satellite dinlas 18*'; battery='*3.5 hour*'; drive='*1 tb*')
Reference	compared to satellite *pallas 21* which can last *4 hour* and has a *500 gb drive* , *satellite dinlas 18* can last *3.5 hour* and has a *1 tb drive* . which one do you prefer
ENCDEC-10	the *satellite pallas 21* has a *500 gb drive* , the *satellite dinlas 18* has a *4 hour battery* life and a *3.5 hour battery* life and a *SLOT_BATTERY* battery life and a *1 tb drive*
HLSTM-10	the *satellite pallas 21* has a *4 hour battery* life and a *500 gb drive* . which one do you prefer [satellite pallas 18] [3.5 hour battery] [1 tb drive]
SCLSTM-10	the *satellite pallas 21* has a *4 hour battery* , and has a *3.5 hour 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 hour battery* life . the *satellite dinlas 18* has a *3.5 hour battery* life and a *SLOT_BATTERY* battery life [1 tb drive]
DualVAE-10	the *satellite pallas 21* has a *4 hour battery* life and a *500 gb drive* and the *satellite dinlas 18* with a *3.5 hour 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 hour battery* and a *SLOT_DRIVE* drive . which one do you prefer [3.5 hour battery]
ENCDEC-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 hour -s* . which one do you prefer [3.5 hour battery]
HLSTM-30	the *satellite pallas 21* is a *500 gb drive* with a *4 hour battery* life . the *satellite dinlas 18* has a *3.5 hour 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 hour battery* life . the *SLOT_NAME* has a *3.5 hour 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 hour battery* life , the *satellite dinlas 18* has a *500 gb drive* and a *3.5 hour 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 hour battery* life while the *satellite dinlas 18* with a *3.5 hour battery* life and a *1 tb drive* . [OK]
CrossVAE-30	the *satellite pallas 21* has a *500 gb drive* with a *4 hour battery* life . the *satellite dinlas 18* has a *1 tb drive* and a *3.5 hour battery* life . which one do you prefer [OK]
CrossVAE-100	the *satellite pallas 21* has a *500 gb drive* with a *4 hour battery* life , while the *satellite dinlas 18* has a *1 tb drive* and a *3.5 hour battery* life . which one do you prefer [OK]

5.6 Conclusion

We have presented in this chapter a Variational-based NLG (VNLG) framework tackling the NLG issues of having a *low-resource* setting data. Based on this framework, we first pro-

pose a novel adversarial VNLG which consists of two critics, domain and text similarity, in an adversarial training procedure, solving the first *domain adaptation* issue. To deal with 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 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 and its variants. The experimental results show that, while the former generator has an ability to perform acceptably well in a new, unseen domain using a limited amount of target domain data, the latter model shows an 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. In the next chapter, we further discuss our main findings in the dissertation as well as directions for future research.

Chapter 6

Conclusions and Future Work

This dissertation has presented a study on applying deep learning techniques for NLG in SDSs. In this chapter, we first give a brief overview of the proposed generators and experimental results. We then summarize the conclusions and key findings, limitations, and point out some directions and outlooks for future studies.

6.1 Conclusions, Key Findings, and Suggestions

The central goal of this dissertation was to deploy DNN-based architectures for NLG in SDSs, addressing essential issues of adequacy, completeness, adaptability and low-resource setting data. Our proposed models in this dissertation mostly address the NLG problems stated in Chapter 1. Moreover, we extensively investigated the effectiveness of the proposed generators in Chapters 3, 4, 5 by training on four different NLG domains and its variants in various scenarios, including scratch, domain adaptation, semi-supervised training with different amount of dataset. It is also worth noting here that all of the proposed generators can learn from unaligned data by jointly training both sentence planning and surface realization to generate natural language utterances. Finally, in addition to the provision of some directions for future research, the dissertation has made following significant contributions to the literature on NLGs for SDSs, since research in such field is still at the early stage of applying deep learning methods, and the related literature is still limited.

Chapter 3 proposed an effective approach to leverage *gating mechanism*, solving the NLG problem in SDSs in terms of adequacy, completeness and a sign of adaptability. We introduced three additional semantic cells into a traditional RNN model to filter the sequential inputs before RNN computations, as well as to select semantic elements and gate a feature vector during generation. The gating generators have not only achieved better performance across all the NLG domains in comparison with the previous gating- and attention-based methods but also obtained highly competitive results compared to a hybrid generator.

In this chapter, the proposed gates are mostly consistent with previous researches (Wen et al., 2015b, 2016a) regarding the ability to effectively control the feature vector DA to drive down the slot error rate. However, there are still some generation cases which consist of consecutive slots (see Figure 4.1) the feature vector DA cannot be adequately controlled. This phenomenon raises a similar problem of the *sentence aggregation*, a subtask of the NLG *sentence planning*, in which the task of sentence aggregation is to combine two or more messages into one sentence. Table 6.1 shows an example of solving sentence aggregation that can generate concise and

better outputs. It is thus important to investigate when the generator should consider sentence aggregation.

Table 6.1: Examples of sentence aggregation in 123 NLG domains

Restaurant DA	inform(name='*Ananda Fuara*'; pricerange='*expensive*'; goodformeal='*lunch*')
Output	*Ananda Fuara* is a nice place, it is in the *expensive price range* and it is *good for lunch*.
Aggre. Output	*Ananda Fuara* is a *good for lunch* place and in the *expensive price range*.
Laptop DA	recommend(name='*Tecra 89*'; type='*laptop*'; platform='*windows 7*'; dimension='*25.4 inch*')
Output	*Tecra 89* is a nice *laptop*. It operates *on windows 7* and its dimensions are *25.4 inch*.
Aggre. Output	*Tecra 89* is a nice *windows 7 laptop* with dimensions of *25.4 inch*.

Chapter 4 proposed a novel *hybrid* NLG framework, which is a combination of gating and attention mechanisms, tackling the NLG problems of *adaptability*, and *adequacy and completeness*. While for the former issue, the proposed models have shown abilities to control the DA vector and quickly scale to a new, unseen domain, the proposed generators for the latter issue have achieved state-of-the-art performances across four NLG domains. The attentional RNN encoder-decoder generation framework mainly consists of three components: an Encoder, an Aligner, and a Decoder, from which two novel generators were proposed. While in the first model, an additional component was introduced by utilizing an idea of attention over attention, a novel decoder was introduced in the second model to select and aggregate semantic elements effectively and to form the required utterances.

In this chapter, one of the key highlights is the introduction of an LSTM-based cell named RALSTM at the decoder side of an encoder-decoder network. It would be worth study to apply the proposed RALSTM cell to other tasks that can be modeled based on the encoder-decoder architecture, *i.e.*, image captioning, reading comprehension, and machine translation. Two follow-up generators (in Chapter 5) on applying the RALSTM model to address NLG problems of *low-resource setting* data have achieved state-of-the-art performances over the previous methods on all training scenarios.

Another key highlight for proposed *gating*-, *attention*- and *hybrid*-based generators in Chapters 3, 4 is that our approaches mainly attack to constrain on an RNN language model as well as decoder component of an encoder-decoder network. While this remains a largely unexplored encoder part in the NLG systems which would be worth investigating in more detail, these conditional language models also have strong potential for straightforward applications in other research areas. Lastly, we found that the proposed model can produce sentences in a correct order than existing generators even though the models are not explicitly designed for the ordering problem. The previous RNN-based generators may have lack of consideration about the order of slot-value pairs during generation. For example, given a DA with pattern: *Compare*(name=*A*, property1=*a1*, property2=*a2*, name=*B*, property1=*b1*, property2=*b2*). The pattern for correct utterances can be: [*A-a1-a2, B-b1-b2*], [*A-a2-a1, B-b2-b1*], [*B-b1-b2, A-a1-a2*], [*B-b1-b2, A-a2-a1*]. Therefore, a generated utterance: "The *A* has *a1* and ***b1*** properties, while the *B* has ***a2*** and *b2* properties" is an incorrect utterance, in which ***b1*** and ***a2*** properties were generated in wrong order. As a result, this occasionally leads to inappropriate sentences. There is thus a need to enhance the ordering problems for NLG as well as other tasks.

Chapter 5 presented two novel variational-based approaches tackling the NLG problems of having a *low-resource* setting data. We first proposed a variational approach for an NLG *domain adaptation* problem, which benefits the generator to adapt faster to a new, unseen domain irrespective of scarce target resources. This model was a combination of a variational generator and two Critics, namely domain and text similarity, in an adversarial training algorithm in

which two critics showed an important role of guiding the model to adapt to a new domain. We then proposed variational neural-based generation model to tackle the NLG problem of having a *low-resource setting in-domain* training dataset. This model was a combination of a variational RNN-RNN generator with a variational CNN-DCNN, in which the proposed models showed an ability to perform acceptably well when the training data is scarce. Moreover, while the variational generator contributes to learning effectively the underlying semantic of DA-utterance pairs, the variational CNN-DCNN showed an important role of encoding useful information into the latent variable.

In this chapter, the proposed variational-based generators show strong performance to tackle the low-resource setting problems, which still leave a large space to further explore regarding some key findings. First, the generators show a good sign to perform the NLG task on the *unsupervised* as well as *semi-supervised* learning. Second, there are potential combinations based on the proposed model terms, such as adversarial training, VAE, autoencoder, encoder-decoder, CNN, DCNN, and so forth. The last potential is that one can think of scenarios to train a multi-domain generator which can simultaneously work well on all existing domains.

In summary, it is also interesting to see in what extent the NLG problems of *completeness*, *adaptability*, and *low-resource setting* are addressed by the generators prosed in previous chapters. For the first issue, all of the proposed generators can effectively solve in case of having sufficient training data in terms of BLEU and slot error rate ERR scores, and in particular, the *variational*-based model which is the current state-of-the-art method. For the *adaptability* issue, while both *gating*- and *hybrid*-based models show a sign of adapting faster to a new domain, the *variational*-based models again demonstrate a strong ability to work acceptably well when there is a modest amount of training data. For the final issue of *low-resource setting data*, while both *gating*- and *hybrid*-based generators have impaired performances, the *variational*-based models can deal with this problem effectively.

6.2 Limitations

Despite the benefits and strengths in solving important NLG issues. There are still some limitations in our work:

- *Dataset bias*: Our proposed models only trained on four original NLG datasets and their variants (see Chapter 2). Despite the fact that these datasets are abundant and diverse enough, it would be better to further assess the effectiveness of the proposed models in a broader range of the other datasets, such as (Lebret et al., 2016; Novikova and Rieser, 2016; Novikova et al., 2017). These datasets introduce additional NLG challenges, such as open vocabulary, complex syntactic structures, and diverse discourse phenomena.

- *Lack of evaluation metrics*: In this dissertation, we only used two evaluation metrics BLEU and slot error rate ERR to examine the proposed models. It would also be better to use more evaluation metrics which bring us a diverse combinatorial assessment of the proposed models, such as NIST (Doddington, 2002), METEOR (Banerjee and Lavie, 2005), ROUGE (Lin, 2004) and CIDER (Vedantam et al., 2015).

- *Lack of human evaluation*: Since there is not always correlation of evaluation between human and automatic metrics, human evaluation provides a more accurate estimation of the systems. However, this process is often expensive and time-consuming.

6.3 Future Work

Based on aforementioned key findings, conclusions, suggestions as well as the limitations, we discuss various lines of research arising from this work which should be pursued.

- Improvement over current models: There are large rooms to enhance the current generators by further investigating into unexplored aspects, such as the encoder component, unsupervised and semi-supervised learning, transfer learning.

- End-to-end trainable dialogue systems: Our proposed models can be easier integrated as an NLG module into an end-to-end task-oriented dialogue systems (Wen et al., 2017b) rather than a non-task-oriented. The latter system often requires a large dataset and views dialogue as a sequence-to-sequence learning (Vinyals and Le, 2015; Zhang et al., 2016; Serban et al., 2016) where the system is trained from a raw source to a raw target sequence. The non-task-oriented is also difficult to evaluate. However, task-oriented dialogue system allows SDS components connect to decide "What to say?" and "How to say it?" in each dialogue terns. Thus, one can leverage the existing models, such as NLG generators, to quickly construct an end-to-end goal-oriented dialogue system.

- Adaptive NLG in SDSs: In our NLG systems, depending on the specific domain, for each meaning representation there may have more than one corresponding response which can be output to the user. Take hotel domain, for example, the dialogue act *inform*(name='*X*'; area='*Y*') might be uttered as "*The X hotel is in the area of Y*" or "*The X is a nice hotel, it is in the Y area*". In the adaptive dialogue system, depending on each *context* NLG should choose the appropriate utterance to output. In the other word, good NLG systems must flexibility adapt their output to the the context. Furthermore, in the case of domain adaptation, the same dialogue act *inform*(name='*X*';area='*Y*') in other domain, *e.g.*, restaurant, the response might also be "*The X restaurant is in the Y area*" or "*The X restaurant is a nice place which is in the Y area*". Thus, good NLG systems must again appropriately adapt the utterances to the changing of context within one domain or even the changing between multi-domain. One can think to train the interactive task-oriented NLG systems by providing additional context to the current training data which is no longer pairs of (dialogue, utterance) but instead triples of (context, dialogue act, utterance).

- Personalized SDSs: Another worthwhile direction for future studies of NLG is to build personalized task-oriented dialogue systems, in which the dialogue systems show an ability to adapt to individual users (Li et al., 2016a; Mo et al., 2017; Mairesse and Walker, 2005). This is an important task, which so far has been mostly untouched (Serban et al., 2015). Personalized dialogue systems allow the target user easier to communicate with the agent and make the dialogue more friendly and efficient. For example, a user (Bob) asks the Coffee machine "*I want a cup of coffee?*", while the non-personalized SDS may response "*Hi there. We have here Espresso, Latte, and Capuccino. What would you want?*", the personalized SDS response more friendly instead "*Hi Bob, still hot Espresso with more sugar?*".

To conclude, we have presented our study on deep learning for NLG in SDSs to tackle some problems of completeness, adaptability, and low-resource setting data. We hope that this dissertation will provide readers useful techniques and inspiration for future research in building much more effective and advanced NLG systems.

Bibliography

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., et al. (2016). Tensorflow: Large-scale machine learning on heterogeneous distributed systems. *arXiv preprint arXiv:1603.04467*.

Angeli, G., Liang, P., and Klein, D. (2010). A simple domain-independent probabilistic approach to generation. In *Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing*, pages 502–512. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Banerjee, S. and Lavie, A. (2005). Meteor: An automatic metric for mt evaluation with improved correlation with human judgments. In *Proceedings of the acl workshop on intrinsic and extrinsic evaluation measures for machine translation and/or summarization*, pages 65–72.

Bangalore, S. and Rambow, O. (2000). Corpus-based lexical choice in natural language generation. In *Proceedings of the 38th Annual Meeting on Association for Computational Linguistics*, pages 464–471. Association for Computational Linguistics.

Barzilay, R. and Lee, L. (2002). Bootstrapping lexical choice via multiple-sequence alignment. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 164–171. Association for Computational Linguistics.

Belz, A. (2005). Corpus-driven generation of weather forecasts. In *Proc. of the 3rd Corpus Linguistics Conference*. Citeseer.

Belz, A., White, M., Van Genabith, J., Hogan, D., and Stent, A. (2010). Finding common ground: Towards a surface realisation shared task. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 268–272. Association for Computational Linguistics.

Bowman, S. R., Vilnis, L., Vinyals, O., Dai, A. M., Józefowicz, R., and Bengio, S. (2015). Generating sentences from a continuous space. *CoRR*, abs/1511.06349.

Busemann, S. and Horacek, H. (1998). A flexible shallow approach to text generation. *arXiv preprint cs/9812018*.

Carenini, G. and Moore, J. D. (2006). Generating and evaluating evaluative arguments. *Artificial Intelligence*, 170(11):925–952.

Chan, W., Jaitly, N., Le, Q., and Vinyals, O. (2016). Listen, attend and spell: A neural network for large vocabulary conversational speech recognition. In *Acoustics, Speech and Signal Processing (ICASSP), 2016 IEEE International Conference on*, pages 4960–4964. IEEE.

Chen, T.-H., Liao, Y.-H., Chuang, C.-Y., Hsu, W. T., Fu, J., and Sun, M. (2017). Show, adapt and tell: Adversarial training of cross-domain image captioner. In *ICCV*.

Cho, K., Van Merriënboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *arXiv preprint arXiv:1409.1259*.

Cui, Y., Chen, Z., Wei, S., Wang, S., Liu, T., and Hu, G. (2016). Attention-over-attention neural networks for reading comprehension. *arXiv preprint arXiv:1607.04423*.

Danlos, L., Meunier, F., and Combet, V. (2011). Easytext: an operational nlg system. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 139–144. Association for Computational Linguistics.

Demberg, V. and Moore, J. D. (2006). Information presentation in spoken dialogue systems. In *11th Conference of the European Chapter of the Association for Computational Linguistics*.

Dethlefs, N. (2017). Domain transfer for deep natural language generation from abstract meaning representations. *IEEE Computational Intelligence Magazine*, 12(3):18–28.

Dethlefs, N., Hastie, H., Cuayáhuitl, H., and Lemon, O. (2013). Conditional random fields for responsive surface realisation using global features. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1254–1263.

Doddington, G. (2002). Automatic evaluation of machine translation quality using n-gram co-occurrence statistics. In *Proceedings of the second international conference on Human Language Technology Research*, pages 138–145. Morgan Kaufmann Publishers Inc.

Duboue, P. A. and McKeown, K. R. (2003). Statistical acquisition of content selection rules for natural language generation. In *Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 121–128. Association for Computational Linguistics.

Dušek, O. and Jurcicek, F. (2015). Training a natural language generator from unaligned data. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 451–461.

Dušek, O. and Jurčíček, F. (2016a). A context-aware natural language generator for dialogue systems. *arXiv preprint arXiv:1608.07076*.

Dušek, O. and Jurčíček, F. (2016b). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

Ganin, Y., Ustinova, E., Ajakan, H., Germain, P., Larochelle, H., Laviolette, F., Marchand, M., and Lempitsky, V. (2016). Domain-adversarial training of neural networks. *Journal of Machine Learning Research*, 17(59):1–35.

Gašić, M., Kim, D., Tsiakoulis, P., and Young, S. (2015). Distributed dialogue policies for multi-domain statistical dialogue management. In *Acoustics, Speech and Signal Processing (ICASSP), 2015 IEEE International Conference on*, pages 5371–5375. IEEE.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*.

Huang, Q., Deng, L., Wu, D., Liu, C., and He, X. (2018). Attentive tensor product learning for language generation and grammar parsing. *arXiv preprint arXiv:1802.07089*.

Inui, K., Tokunaga, T., and Tanaka, H. (1992). Text revision: A model and its implementation. In *Aspects of automated natural language generation*, pages 215–230. Springer.

Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference CVPR*, pages 3128–3137.

Keizer, S. and Rieser, V. (2018). Towards learning transferable conversational skills using multi-dimensional dialogue modelling. *arXiv preprint arXiv:1804.00146*.

Kingma, D. P. and Welling, M. (2013). Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*.

Kondadadi, R., Howald, B., and Schilder, F. (2013). A statistical nlg framework for aggregated planning and realization. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1406–1415.

Konstas, I. and Lapata, M. (2013). A global model for concept-to-text generation. *J. Artif. Intell. Res.(JAIR)*, 48:305–346.

Langkilde, I. (2000). Forest-based statistical sentence generation. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, pages 170–177. Association for Computational Linguistics.

Langkilde, I. and Knight, K. (1998). Generation that exploits corpus-based statistical knowledge. In *Proceedings of the 36th Annual Meeting of the Association for Computational Linguistics and 17th International Conference on Computational Linguistics-Volume 1*, pages 704–710. Association for Computational Linguistics.

Langkilde-Geary, I. (2002). An empirical verification of coverage and correctness for a general-purpose sentence generator. In *Proceedings of the international natural language generation conference*, pages 17–24.

Lebret, R., Grangier, D., and Auli, M. (2016). Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Li, J., Galley, M., Brockett, C., Gao, J., and Dolan, B. (2015). A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055*.

Li, J., Galley, M., Brockett, C., Spithourakis, G. P., Gao, J., and Dolan, B. (2016a). A persona-based neural conversation model. *arXiv preprint arXiv:1603.06155*.

Li, J. and Jurafsky, D. (2016). Mutual information and diverse decoding improve neural machine translation. *arXiv preprint arXiv:1601.00372*.

Li, J., Monroe, W., Ritter, A., Galley, M., Gao, J., and Jurafsky, D. (2016b). Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*.

Lin, C.-Y. (2004). Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Lu, J., Xiong, C., Parikh, D., and Socher, R. (2016). Knowing when to look: Adaptive attention via a visual sentinel for image captioning. *arXiv preprint arXiv:1612.01887*.

Luong, M.-T., Le, Q. V., Sutskever, I., Vinyals, O., and Kaiser, L. (2015a). Multi-task sequence to sequence learning. *arXiv preprint arXiv:1511.06114*.

Luong, M.-T., Pham, H., and Manning, C. D. (2015b). Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*.

Mairesse, F., Gašić, M., Jurčíček, F., Keizer, S., Thomson, B., Yu, K., and 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*, ACL '10, pages 1552–1561, Stroudsburg, PA, USA. Association for Computational Linguistics.

Mairesse, F. and Walker, M. (2005). Learning to personalize spoken generation for dialogue systems. In *Ninth European Conference on Speech Communication and Technology*.

Mairesse, F. and Young, S. (2014). Stochastic language generation in dialogue using factored language models. *Computational Linguistics*.

Marsi, E. C. (2001). *Intonation in spoken language generation*. PhD thesis, Radboud University Nijmegen.

Mathur, P., Ueffing, N., and Leusch, G. (2018). Multi-lingual neural title generation for e-commerce browse pages. *arXiv preprint arXiv:1804.01041*.

McRoy, S. W., Channarukul, S., and Ali, S. S. (2000). Yag: A template-based generator for real-time systems. In *Proceedings of the first international conference on Natural language generation-Volume 14*, pages 264–267. Association for Computational Linguistics.

McRoy, S. W., Channarukul, S., and Ali, S. S. (2001). Creating natural language ouput for real-time applications. *intelligence*, 12(2):21–34.

Mei, H., Bansal, M., and Walter, M. R. (2015). What to talk about and how? selective generation using lstms with coarse-to-fine alignment. *arXiv preprint arXiv:1509.00838*.

Meteer, M. W. (1991). Bridging the generation gap between text planning and linguistic realization. *Computational Intelligence*, 7(4):296–304.

Mikolov, T. (2010). Recurrent neural network based language model. In *INTERSPEECH*.

Mo, K., Zhang, Y., Yang, Q., and Fung, P. (2017). Fine grained knowledge transfer for personalized task-oriented dialogue systems. *arXiv preprint arXiv:1711.04079*.

Mrkšić, N., Séaghdha, D. O., Thomson, B., Gašić, M., Su, P.-H., Vandyke, D., Wen, T.-H., and Young, S. (2015). Multi-domain dialog state tracking using recurrent neural networks. *arXiv preprint arXiv:1506.07190*.

Nallapati, R., Zhou, B., Gulcehre, C., Xiang, B., et al. (2016). Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*.

Neculoiu, P., Versteegh, M., Rotaru, M., and Amsterdam, T. B. (2016). Learning text similarity with siamese recurrent networks. *ACL 2016*, page 148.

Novikova, J., Dušek, O., and Rieser, V. (2017). The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual Meeting of the Special Interest Group on Discourse and Dialogue*, Saarbrücken, Germany. arXiv:1706.09254.

Novikova, J. and Rieser, V. (2016). The analogue challenge: Non aligned language generation. In *Proceedings of the 9th International Natural Language Generation conference*, pages 168–170.

Oh, A. H. and Rudnicky, A. I. (2000). Stochastic language generation for spoken dialogue systems. In *Proceedings of the 2000 ANLP/NAACL Workshop on Conversational systems-Volume 3*, pages 27–32. Association for Computational Linguistics.

Oliver, J. M. M. E. F. and White, L. M. (2004). Generating tailored, comparative descriptions in spoken dialogue. *AAAI*.

Paiva, D. S. and Evans, R. (2005). Empirically-based control of natural language generation. In *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*, pages 58–65.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th ACL*, pages 311–318. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *EMNLP*, volume 14, pages 1532–43.

Radford, A., Metz, L., and Chintala, S. (2015). Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rambow, O., Bangalore, S., and Walker, M. (2001). Natural language generation in dialog systems. In *Proceedings of the first international conference on Human language technology research*, pages 1–4. Association for Computational Linguistics.

Ratnaparkhi, A. (2000). Trainable methods for surface natural language generation. In *Proceedings of the 1st NAACL*, pages 194–201. Association for Computational Linguistics.

Reiter, E., Dale, R., and Feng, Z. (2000). *Building natural language generation systems*, volume 33. MIT Press.

Reiter, E., Sripada, S., Hunter, J., Yu, J., and Davy, I. (2005). Choosing words in computer-generated weather forecasts. *Artificial Intelligence*, 167(1-2):137–169.

Rieser, V., Lemon, O., and Liu, X. (2010). Optimising information presentation for spoken dialogue systems. In *Proceedings of the 48th ACL*, pages 1009–1018. Association for Computational Linguistics.

Rush, A. M., Chopra, S., and Weston, J. (2015). A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*.

Serban, I. V., Lowe, R., Henderson, P., Charlin, L., and Pineau, J. (2015). A survey of available corpora for building data-driven dialogue systems. *arXiv preprint arXiv:1512.05742*.

Serban, I. V., Sordoni, A., Bengio, Y., Courville, A. C., and Pineau, J. (2016). Building end-to-end dialogue systems using generative hierarchical neural network models. In *AAAI*, volume 16, pages 3776–3784.

Siddharthan, A. (2010). Complex lexico-syntactic reformulation of sentences using typed dependency representations. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 125–133. Association for Computational Linguistics.

Stent, A., Prasad, R., and Walker, M. (2004). Trainable sentence planning for complex information presentation in spoken dialog systems. In *Proceedings of the 42nd ACL*, page 79. Association for Computational Linguistics.

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

Tran, V. K. and Nguyen, L. M. (2017b). Semantic refinement gru-based neural language generation for spoken dialogue systems. In *15th International Conference of the Pacific Association for Computational Linguistics, PACLING 2017*, Yangon, Myanmar.

Tran, V. K. and Nguyen, L. M. (2018a). Adversarial domain adaptation for variational natural language generation in dialogue systems. In *COLING.*, pages 1205–1217, Santa Fe, New Mexico, USA.

Tran, V. K. and Nguyen, L. M. (2018b). Dual latent variable model for low-resource natural language generation in dialogue systems. In *ConLL. Accepted*, Brussels, Belgium.

Tran, V. K. and Nguyen, L. M. (2018c). Encoder-decoder recurrent neural networks for natural language genration in dialouge systems. *Transactions on Asian and Low-Resource Language Information Processing (TALLIP). Submitted.*

Tran, V. K. and Nguyen, L. M. (2018d). Gating mechanism based natural language generation for spoken dialogue systems. *Neurocomputing. Submitted.*

Tran, V. K. and Nguyen, L. M. (2018e). Variational model for low-resource natural language generation in spoken dialogue systems. *Journal of Computer Speech and Language. Submitted.*

Tran, V. K., Nguyen, L. M., and Tojo, S. (2017a). Neural-based natural language generation in dialogue using rnn encoder-decoder with semantic aggregation. In *Proceedings of the 18th Annual Meeting on Discourse and Dialogue, SIGDIAL 2017*, pages 231–240, Saarbrücken, Germany. Association for Computational Linguistics.

Tran, V. K., Nguyen, V. T., Shirai, K., and Nguyen, L. M. (2017b). Towards domain adaptation for neural network language generation in dialogue. In *4th NAFOSTED Conference on Information and Computer Science, NICS 2017*, pages 19–24.

Vedantam, R., Lawrence Zitnick, C., and Parikh, D. (2015). Cider: Consensus-based image description evaluation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4566–4575.

Vinyals, O. and Le, Q. (2015). A neural conversational model. *arXiv preprint arXiv:1506.05869*.

Vinyals, O., Toshev, A., Bengio, S., and Erhan, D. (2015). Show and tell: A neural image caption generator. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3156–3164.

Walker, M. A., Rambow, O., and Rogati, M. (2001). Spot: A trainable sentence planner. In *Proceedings of the second meeting of the North American Chapter of the Association for Computational Linguistics on Language technologies*, pages 1–8. Association for Computational Linguistics.

Walker, M. A., Stent, A., Mairesse, F., and Prasad, R. (2007). Individual and domain adaptation in sentence planning for dialogue. *Journal of Artificial Intelligence Research*, 30:413–456.

Walker, M. A., Whittaker, S. J., Stent, A., Maloor, P., Moore, J., Johnston, M., and Vasireddy, G. (2004). Generation and evaluation of user tailored responses in multimodal dialogue. *Cognitive Science*, 28(5):811–840.

Wang, B., Liu, K., and Zhao, J. (2016). Inner attention based recurrent neural networks for answer selection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*.

Wen, T.-H., Gašić, M., Kim, D., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015a). 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., and Young, S. (2016a). Multi-domain neural network language generation for spoken dialogue systems. *arXiv preprint arXiv:1603.01232*.

Wen, T.-H., Gašic, M., Mrkšic, N., Rojas-Barahona, L. M., Su, P.-H., Vandyke, D., and Young, S. (2016b). Toward multi-domain language generation using recurrent neural networks. *NIPS Workshop on ML for SLU and Interaction*.

Wen, T.-H., Gašić, M., Mrkšić, N., Su, P.-H., Vandyke, D., and Young, S. (2015b). 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., and Young, S. (2017a). Latent intention dialogue models. *arXiv preprint arXiv:1705.10229*.

Wen, T.-H., Vandyke, D., Mrkšić, N., Gasic, M., Rojas Barahona, L. M., Su, P.-H., Ultes, S., and Young, S. (2017b). A network-based end-to-end trainable task-oriented dialogue system. In *EACL*, pages 438–449, Valencia, Spain. Association for Computational Linguistics.

Werbos, P. J. (1990). Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

Williams, J. (2013). Multi-domain learning and generalization in dialog state tracking. In *Proceedings of SIGDIAL*, volume 62. Citeseer.

Williams, S. and Reiter, E. (2005). Generating readable texts for readers with low basic skills. In *Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05)*.

Wong, Y. W. and Mooney, R. (2007). Generation by inverting a semantic parser that uses statistical machine translation. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proceedings of the Main Conference*, pages 172–179.

Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., et al. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*.

Xu, K., Ba, J., Kiros, R., Cho, K., Courville, A. C., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *ICML*, volume 14, pages 77–81.

Yang, Z., Yuan, Y., Wu, Y., Cohen, W. W., and Salakhutdinov, R. R. (2016). Review networks for caption generation. In *Advances in Neural Information Processing Systems*, pages 2361–2369.

You, Q., Jin, H., Wang, Z., Fang, C., and Luo, J. (2016). Image captioning with semantic attention. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4651–4659.

Young, S., Gašić, M., Keizer, S., Mairesse, F., Schatzmann, J., Thomson, B., and Yu, K. (2010). The hidden information state model: A practical framework for pomdp-based spoken dialogue management. *Computer Speech & Language*, 24(2):150–174.

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

Zhang, X. and Lapata, M. (2014). Chinese poetry generation with recurrent neural networks. In *EMNLP*, pages 670–680.

Publications

Journals

[1] Van-Khanh Tran, Le-Minh Nguyen, **Gating Mechanism based Natural Language Generation for Spoken Dialogue Systems**, *submitted to* Journal of Neurocomputing, May 2018.

[2] Van-Khanh Tran, Le-Minh Nguyen, **Encoder-Decoder Recurrent Neural Networks for Natural Language Genration in Dialouge Systems**, *submitted to* journal Transactions on Asian and Low-Resource Language Information Processing (TALLIP), August 2018.

[3] Van-Khanh Tran, Le-Minh Nguyen, **Variational Model for Low-Resource Natural Language Generation in Spoken Dialogue Systems**, *submitted to* Journal of Computer Speech and Language, August 2018.

International Conferences

[4] Van-Khanh Tran, Le-Minh Nguyen, **Adversarial Domain Adaptation for Variational Natural Language Generation in Dialogue Systems**, *Accepted at* The 27th International Conference on Computational Linguistics (COLING), pp. 1205-1217, August 2018. Santa Fe, New-Mexico, USA.

[5] Van-Khanh Tran, Le-Minh Nguyen, **Dual Latent Variable Model for Low-Resource Natural Language Generation in Dialogue Systems**, *Accepted at* The 22nd Conference on Computational Natural Language Learning (CoNLL), November 2018. Brussels, Belgium.

[6] Van-Khanh Tran, Le-Minh Nguyen, **Natural Language Generation for Spoken Dialogue System using RNN Encoder-Decoder Network**, Proceedings of the 21st Conference on Computational Natural Language Learning (CoNLL), pp. 442-451, August 2017. Vancouver, Canada.

[7] Van-Khanh Tran, Le-Minh Nguyen, Tojo Satoshi, **Neural-based Natural Language Generation in Dialogue using RNN Encoder-Decoder with Semantic Aggregation**, Proceedings of the 18th Annual Meeting on Discourse and Dialogue (SIGDIAL), pp. 231-240, August 2017. Saarbrücken, Germany.

[8] Van-Khanh Tran, Le-Minh Nguyen, **Semantic Refinement GRU-based Neural Language Generation for Spoken Dialogue Systems**, The 15th International Conference of the Pacific Association for Computational Linguistics (PACLING), pp. 63–75, August 2017. Yangon, Myanmar.

[9] Van-Khanh Tran, Van-Tao Nguyen, Le-Minh Nguyen, **Enhanced Semantic Refinement Gate for RNN-based Neural Language Generator**, The 9th International Conference on Knowledge and Systems Engineering (KSE), pp. 172-178, October 2017. Hue, Vietnam.

[10] Van-Khanh Tran, Van-Tao Nguyen, Kiyoaki Shirai, Le-Minh Nguyen, **Towards Domain Adaptation for Neural Network Language Generation in Dialogue**, The 4th NAFOSTED Conference on Information and Computer Science (NICS), pp. 19-24, August 2017. Hanoi, Vietnam.

International Workshops

[11] S. Danilo Carvalho, Duc-Vu Tran, Van-Khanh Tran, Le-Minh Nguyen, **Improving Legal Information Retrieval by Distributional Composition with Term Order Probabilities**, Competition on Legal Information Extraction/Entailment (COLIEE), March 2017.

[12] S. Danilo Carvalho, Duc-Vu Tran, Van-Khanh Tran, Dac-Viet Lai, Le-Minh Nguyen, **Lexical to Discourse-Level Corpus Modeling for Legal Question Answering**, Competition on Legal Information Extraction/Entailment (COLIEE), February 2016.

Awards

- Best Student Paper Award at The 9th International Conference on Knowledge and Systems Engineering (KSE), October 2017. Hue, Vietnam.