Searching Optimal Solutions for Sequence-to-Sequence Models

A Thesis

Presented to

the faculty of the School of Engineering and Applied Science

University of Virginia

---

in partial fulfillment
of the requirements for the degree

Master of Science

by

Wanyu Du

May 2020

APPROVAL SHEET

This Thesis
is submitted in partial fulfillment of the requirements
for the degree of
Master of Science

Author Signature: Wanyu Du

This Thesis has been read and approved by the examining committee:

Advisor: Yangfeng Ji

Committee Member: Yanjun Qi

Committee Member: Hongning Wang

Committee Member: _____

Committee Member: _____

Committee Member: _____

Accepted for the School of Engineering and Applied Science:

Craig H. Benson, School of Engineering and Applied Science

May 2020

ACKNOWLEDGEMENTS

First, I would like to thank my advisor Professor Yangfeng Ji. The door to Professor Ji's office was always open whenever I had questions about my projects. His consistent support steered me in the right direction and made me achieve solid results. Under his supervision, I started my first step in academic research in the field of computer science. I really appreciate his unreserved support.

I would also like to thank my thesis committee members: Professor Yanjun Qi and Professor Hongning Wang for their precious advice and comments on my work. Their advice guides me to further polish my thesis, making it becomes a more solid and comprehensive work.

Finally, I want to express my gratitude to my parents and to my grandma for providing me with continuous encouragement and support throughout my life. This accomplishment would not be possible without them. Thank you and love you all!

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

SUMMARY

Sequence-to-sequence generation applications take source texts as inputs, and automatically generate new texts that satisfy specific target requirements, such as generating a paraphrase, translating to another language, answering a question, etc. There are two key challenges in sequence-to-sequence generation applications: first, how to encode source texts into informative representations that preserve rich semantic information; second, how to generate target texts that look like human-generated texts. In this thesis, I develop probabilistic models to encode informative context representations from source texts using variational autoencoders, and investigate different learning algorithms to train models that can effectively generate better target texts.

For learning context representations with variational autoencoders, I identify the limitation of using variational autoencoders for sequence-to-sequence models is that applying the standard normal prior is likely to trap the variational posterior into local optimal, thus preventing the model from learning rich context representations. Therefore, I propose to adapt the attention mechanism and learn some empirical priors to help the model get rid of the local optimal and learn better context representations.

For investigating different learning algorithms for sequence-to-sequence models, I present an empirical study on different learning algorithms (e.g. REINFORCE, DAGGER) to analyze how they can the training-inference discrepancy when training sequence-to-sequence models. I apply different learning algorithms in state-of-the-art model [1] in paraphrase generation tasks, and find that DAGGER constantly contributes to better performance.

CHAPTER 1
INTRODUCTION

Sequence-to-sequence generation is an important subfield of Natural Language Processing (NLP) which has a wide range of applications, such as paraphrase generation [2], dialogue system [3], machine translation [4], question answering [5], etc. To learn an optimal policy for sequence-to-sequence generation tasks, the model needs to generate sentence that not only preserves the semantic meaning of source sentence, but also has rich linguistic styles that look like human generated texts. There are two major aspects contributing to getting the optimal policy in sequence-to-sequence generation: (1) encoding rich source content information; and (2) generating correct and diversified target sentences.

Although prior works developed various methods [6, 7, 8] in sequence-to-sequence generation, the recent progress mainly applies neural networks [9] to encode source information and generate target sentence. Particularly, the encoder-decoder framework [10] is widely adopted, in this work, I define it as Sequence-to-Sequence (Seq2Seq) framework. As shown in Figure 1.1, the encoder is a recurrent neural network (RNN) which encodes the input sentence into a deterministic latent representation that captures the semantic information of the source sentence. The decoder is another RNN which generates a word at each time step conditioning on previous words and the encoded latent representations. Note that the input word to the decoder during training is sampled from the ground truth data distribution, while it switches to the model prediction distribution during testing, since the ground truth data is no longer available. This training strategy is called *teacher forcing* [11], which is a widely used learning algorithm for training recurrent neural networks.

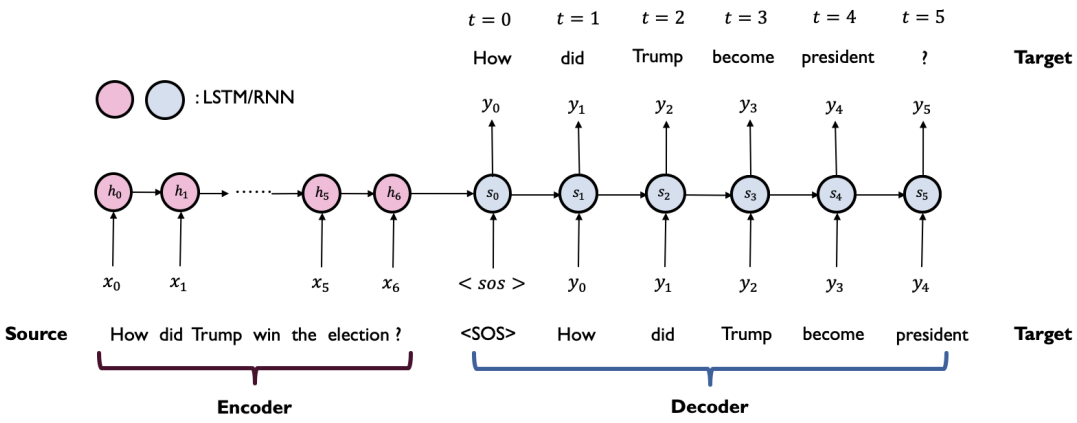

Figure 1.1: The general Seq2Seq framework for sequence-to-sequence generation.

1.1 Problems in Sequence-to-Sequence Generation

There are two major problems under this Seq2Seq framework: (1) the deterministic latent representation prevents the decoder from generating diversified texts; (2) the training-inference discrepancy caused by teacher forcing makes the decoder accumulate and propagate errors when generating texts. The first problem can be identified as learning diversified and informative latent representations for texts. In this work, I resort to apply probabilistic models which learn a probability distribution of the latent representation. By sampling the latent representation from a probabilistic distribution, the decoder is able to generate diversified predictions that can be applied in many sequence-to-sequence generation tasks, such as text style transfer [12], paraphrase generation [13], etc. The second problem can be described as finding more suitable learning algorithm for Seq2Seq models. In this work, I conduct an empirical comparison on reinforcement learning [14] and imitation learning [15] algorithms to analyze how they can alleviate the training-inference discrepancy problem in Seq2Seq models.

1.2 Probabilistic Representation Learning for Seq2Seq Models

Given the observed data $\boldsymbol{x}$, $\boldsymbol{z}$ is the latent variable of $\boldsymbol{x}$, the probability distribution of latent variable is defined as:

$$p(\boldsymbol{z}|\boldsymbol{x}) = \frac{\prod_{i=1}^{M} p(\boldsymbol{x}^{(i)}, \boldsymbol{z}^{(i)})}{\prod_{i=1}^{M} p(\boldsymbol{x}^{(i)})} \tag{1.1}$$

Since the marginal data distribution $p(\boldsymbol{X}) = \prod_{i=1}^{M} p(\boldsymbol{x}^{(i)})$ is not tractable in practice, we use variational inference [16] to find an alternative distribution $q(\boldsymbol{z}|\boldsymbol{x})$ as an approximation to the true posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$. A typical framework for vairational inference with deep neural networks is variational autoencoders [17, 18].

Variational autoencoders apply an inference network to model the approximated posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$, also called the variational posterior, using amortized inference [19]. Each input sentence $\boldsymbol{x}^{(i)}$ shares the same set of parameters ϕ, and different input sentence $\boldsymbol{x}^{(i)}$ will lead to different variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$. Variational autoencoders use a generative model $p_\theta(\boldsymbol{x}|\boldsymbol{z}^{(i)})$ to reconstruct the input sentence $\boldsymbol{x}^{(i)}$, where $\boldsymbol{z}^{(i)}$ is sampled from the variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$. If the variational posterior learns a good approximation to the true posterior distribution $p(\boldsymbol{z}|\boldsymbol{x}^{(i)})$, the generative model will be able to reconstruct high quality input sequence $\boldsymbol{x}^{(i)}$.

Previous work [20] shows that latent representations of real-world text data are often multi-modal and highly complex. In practice, variational autoencoders often fail to learn a good approximation to the true latent manifold, and generate random text which preserve little information from the input text. This failure comes from a bad local optimal [21], where the variational posterior always equals to the prior and the model learns nothing in the latent representations. The bad local optimal happens because the Kullback-Leibler (KL) divergence term in loss function encourages the variational posterior to match the prior. Since the prior in variational autoencoders is usually a standard normal distribution, the variational posterior will eventually collapse to the standard normal distribution, making variational autoencoders encode just random noise into the latent representations.

There has been some previous works proposing to learn an appropriate prior to help variational autoencoders better approximate the true latent manifold. [22] suggests to approximate the optimal prior by assembling a mixture of Gaussian posteriors with pseudo-inputs learned during training. [23] proposes to learn a Real Non Volume Preserving (RNVP) prior to improve the performance of variational autoencoders. Other existing works replace the Gaussian distributions with more complicated distributions to fit variational autoencoders on text data. [20] uses a piecewise constant probability density function to parameterize latent variable $\boldsymbol{z}$, but this distribution only serves for the topic modeling task. [21] applies the von Mises-Fisher (vMF) distribution for variational autoencoders, but this distribution gains small performance improvement compared with Gaussian distribution.

In chapter 3, I propose to apply Gaussian Process (GP) priors to capture the local latent manifold of source text $\boldsymbol{x}$, in order to help variational autoencoders better approximate the true latent manifold. Gaussian Process is often used for exact inference [24, 25, 26], but some prior work [27] also applies it as an extension of the variational autoencoders where correlations between data samples are modeled through a GP prior on latent variables. I follow the prior work [27] and propose a simple GP prior which fits into the variational Seq2Seq framework.

1.3 Efficient Learning Algorithms for Seq2Seq Models

The most widely-used learning algorithm for training the RNN decoder is maximum likelihood estimation (MLE), which is also called *teacher forcing* [11]. During training, the decoder makes prediction conditioning on the ground truth word at previous time step. While during testing, the ground truth word is no longer available, the decoder makes prediction conditioning on the word sampled from the model distribution at previous time step.

Previous work [28] argues that *teacher forcing* results in poor prediction performance of the decoder. Since the word sampled from the model distribution at previous time step

may be very different from the word sampled from the ground truth data distribution, and the decoder is never trained to make prediction conditioning on the words sampled from the model distribution.

To address this challenge, prior work [13] suggests to utilize the exploration strategy in reinforcement learning (RL). A typical way [29, 30, 31] of using RL is to use the word sampled from the probability distribution at previous time step as the input to the decoder during training, and use the ground truth text to compute the reward for updating the current policy. However, training with RL algorithms is not trivial and often hardly works in practice [32]. Another category in the middle ground between RL and MLE is imitation learning (IL) [33, 34], which mixes the two kinds of inputs to the decoder during training, and directly uses the ground truth text to update the current policy.

In chapter 4, I present an empirical comparison between different learning algorithms (i.e. MLE, RL and IL) to demonstrate the pros and cons of using them for training RNNs. I propose a unified framework to include some popular learning algorithms as special cases, such as the REINFORCE algorithm [35] in RL and the DAGGER algorithm [34] in IL. To better understand the value of different training strategies, I further propose several variant learning algorithms based on the RL framework. Experiments on the benchmark datasets show that the DAGGER algorithm helps the decoder making better predictions than the REINFORCE algorithm and its variants in paraphrase generation.

1.4 Contributions

To sum up, with the goal of jumping out of local optimal and searching better solutions for Seq2Seq models, I explore probabilistic models for representation learning and efficient learning algorithms in various sequence-to-sequence generation tasks.

For probabilistic representation learning, I analyze the local geometry of latent manifolds of source texts, and propose to use variational Seq2Seq models with Gaussian Process priors to better approximate the true latent manifolds of Seq2Seq models. The experiment

results show that Gaussian Process priors can help variational Seq2Seq models outperform traditional Seq2Seq models in both paraphrase generation and style transfer tasks.

For efficient learning algorithms, I build a unified learning framework for popular learning algorithms (i.e. MLE, RL and IL), and conduct empirical experiments to compare the benefits and limitations of different learning algorithms in paraphrase generation. The experiment results show that IL is constantly better than RL and MLE for training Seq2Seq models, and can outperform the state-of-the-art model with a large margin.

CHAPTER 2
RELATED WORKS

This chapter summarizes prior works in probabilistic representation learning for texts and efficient learning algorithms for training recurrent neural networks. The discussion on previous works can provide both theoretical insights and empirical observations of challenges in sequence-to-sequence generation.

2.1 Probabilistic Representation Learning for Seq2Seq Models

With the goal of learning diversified and informative latent representations of real-world data, some previous works [17, 18, 36] choose to apply deep latent variable models, which are good at learning rich, non-linear data representations and also specify prior knowledge to allow uncertainty about unknown factors. Variational autoencoders [17, 18] are the general framework for learning deep latent variable models, where they apply amortized variational inference [37, 38] to approximate the true posterior distribution of latent variables.

Despite of their solid mathematical basis from Bayesian inference [39], variational autoencoders often suffer from poor generative performance. [40] claims that the poor performance of variational autoencoders comes the mismatch between the true and approximated posterior distribution of latent variables, which they refer to as the *inference gap*. They decompose the *inference gap* into two components: the *approximation gap* which comes from the inability of the variational distribution family to exactly match the true posterior, and the *amortization gap* which comes from amortizing the variational parameters over the entire training set instead of learning variational parameters for each training example individually. By comparing the *inference gap* between variational autoencoders with different variational distribution families, they find that the *amortization gap* is the major cause of

the inference sub-optimal in variational autoencoders, because the decoder can learn to accommodate different choice of variational distribution family. Besides, they show that improving the expressiveness of approximated posteriors can effectively reduce the *amortization gap*, thus making variational autoencoders achieve much better performance.

There have been extensive prior works on improving the expressiveness of approximated posteriors in the field of computer vision. [19, 41] uses more flexible approximated posteriors (e.g, normalizing flow, auto-regressive flow). [42, 43] combines the amortized variational inference with stochastic variational inference to further refine the approximated posterior distribution. Another group of works propose that learning flexible priors is beneficial to improving the expressiveness of approximated posteriors. [44] demonstrates that updating the prior is helpful for both sample generation and inference, and propose to use invertible functions to parameterize explicit densities for both the prior and the variational posterior. [22] uses a mixture of variational posteriors as the prior, and empirically show that applying the proposed prior can successfully increase the generative performance of variational autoencoders.

In the context of sequence-to-sequence generation applications, some previous works suggest to replace the Gaussian distribution family with other distribution families which will be more suitable to model complex text data. [20] demonstrates that variational autoencoders with the standard Gaussian prior is incapable of representing complex latent factors efficiently in the natural language text, and apply a piece-wise constant distribution as their prior. [21] replaces the Gaussian distribution with von Mises-Fisher (vMF) distribution, and manage to achieve better performance in document modeling and language modeling tasks. The existing works mainly focus on applying different variational distribution families, few works have investigated how to improve the expressiveness of approximated posteriors for text data.

In this work, I propose to learn empirical Gaussian Process priors to help improve the expressiveness of approximated posteriors for text data in variational autoencoders.

The proposed Gaussian Process prior is an ad-hoc prior which only applies to the current encoder hidden state, which enables the modeling of multi-modal distribution of text data in the latent space. By conducting empirical experiments on some sequence-to-sequence generation applications (i.e. paraphrase generation, style transfer), I show that the proposed Gaussian Process priors can effectively help variational autoencoders better approximate the true posterior distribution, thus achieving better performance than previous works.

2.2 Efficient Learning Algorithms for Seq2Seq Models

In supervised learning, a well-known challenge of training recurrent neural networks is the exposure bias problem [29]: the current prediction of the generative model conditions on the ground truth words during training, but switches to conditioning on previous predictions during testing, causing the model accumulating and propagating errors when generating the text.

Prior works [30, 31] follow the training strategy of reinforcement learning [14] for alleviating the exposure bias problem, where they introduce previous predictions as the input to recurrent neural networks during training. [30] lets the model generate current prediction conditioning on previous predictions during training, and propose to use minimum risk training to minimize the discrepancy between model predictions and ground truth sentence, where the discrepancy is computed by evaluation metrics for the sentence generation task (e.g. BLEU[45], METEOR[46], etc.). [31] shares similar idea, but use actor-critic methods to alleviate the discrepancy between training and testing. They train a critic network to compute the discrepancy between the model predictions and ground truth sentence, and optimize the discrepancy in order to learn a better policy.

Other work [15] follows the training strategy of imitation learning [33, 34], and randomly introduce previous predictions as the input to recurrent neural with some probability. This method is called *scheduled sampling*, different from reinforcement learning, it uses the ground truth sentence as the expert action to update its current policy. In other

words, it directly maximizes the log-probability of ground truth sentence during training, and does not require to compute the discrepancy between model predictions and ground truth sentence to update its current policy.

Another work [28] applies the adversarial domain adaptation [47, 48] to train a discriminator to discriminate between predictions distribution and ground truth distribution. This method is called *professor forcing*, by training the discriminator jointly with the generative model, it aims at making the generative model produce predictions distribution that is very similar to ground truth distribution.

While both reinforcement learning and imitation learning are widely used in sequential prediction tasks, there is a lack of direct comparison between the two different types of training strategies, which leads to only a partial image on their benefits. In this work, I conduct an empirical study on how reinforcement learning and imitation learning can help alleviate the exposure bias problem in paraphrase generation. Experiments on benchmark datasets show that imitation learning is constantly better than reinforcement learning, and training the state-of-the-art model [1] with imitation learning can help gain a further performance improvement.

CHAPTER 3

PROBABILISTIC REPRESENTATION LEARNING FOR SEQ2SEQ MODELS

This chapter discusses how to use variational inference and deep neural networks to learn diversified and informative latent representations in Seq2Seq models.

Many NLP applications aim at transforming source text information into target text information, such as machine translation [4], paraphrase generation [9], style transfer [49] etc. In this chapter, I extend the variational autoencoders framework to varitaional Seq2Seq framework, in order to fit it into a wider range of sequence-to-sequence generation applications. However, this vanilla adaptation often leads to a bad local optimal [50], where the variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ collapses to the standard normal prior, and the RNN decoder degrades to a RNN language model which only relies on the previous target word to make current prediction during training.

To avoid being trapped in the bad local optimal, previous works [50, 51] suggest to introduce attention mechanism [4] to enable the decoder to adaptively incorporate the information from source sequences. In this chapter, I explored different ways to adopt attention mechanism into the variational Seq2Seq framework. In addition, I propose to learn some flexible priors to further prevent variational Seq2Seq models from being trapped in bad local optimal. Besides, I propose to conduct local geometric analysis on latent representations to analyze whether variational Seq2Seq models learn a good approximation to the true low-dimensional latent manifold of source sequences.

The rest of this chapter is organized as follows: section 3.1 introduces the variational Seq2Seq framework and explains why the model is trapped in the bad local optimal during optimization; section 3.2 discusses how to model the attention vectors as latent random variables in the variational Seq2Seq framework; section 3.3 introduces how to model the source encodings as latent random variables in the variational Seq2Seq framework; sec-

tion 3.4 introduces how to do local geometric analysis on latent representations to analyze whether variational Seq2Seq models learn a good approximation to the true latent manifold of source sequences; section 3.5 defines evaluation metrics and shows empirical experiment results for different variational Seq2Seq models; section 3.6 concludes the most effective method to learn diversified and informative latent representations in Seq2Seq models.

3.1 Variational Seq2Seq Models

In order to extend the variational autoencoder to the variational Seq2Seq models (also called varitional encoder-decoders), one possible way [52, 3] is to condition probabilistic distributions on both source sequence $\boldsymbol{x}$ and target sequence $\boldsymbol{y}$, such that the variational posterior is $q_\phi(\boldsymbol{z}|\boldsymbol{x}, \boldsymbol{y})$. However, this introudces a discrepancy between training and prediction, since $\boldsymbol{y}$ is not available during prediction. Another widely used way [53, 54, 50] is to condition latent variables $\boldsymbol{z}$ only on source sequence $\boldsymbol{x}$. They assume that target sequence $\boldsymbol{y}$ is a function of source sequence $\boldsymbol{x}$, so that they have $q_\phi(\boldsymbol{z}|\boldsymbol{y}) = q_\phi(\boldsymbol{z}|f(\boldsymbol{x})) \triangleq q_\phi(\boldsymbol{z}|\boldsymbol{x})$. In this work, I follow the second approach [53, 50] to build our variational Seq2Seq framework.

Let words from the source sequence be $\boldsymbol{x} = \{\boldsymbol{x}_1, \boldsymbol{x}_2, ..., \boldsymbol{x}_N\}$, words from the target sequence be $\boldsymbol{y} = \{\boldsymbol{y}_1, \boldsymbol{y}_2, ..., \boldsymbol{y}_T\}$, and the latent variable be $\boldsymbol{z}$. For the Seq2Seq model, both the encoder and decoder are recurrent neural networks (RNNs). The RNN encoder encodes the source sequence $\boldsymbol{x}$ into a series of hidden states $\boldsymbol{h} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N\}$. For the standard variationl Seq2Seq framework, the last hidden state $\boldsymbol{h}_N$ is passed to a recognition model to model the variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ using amortized inference [17]. Note that the latent variable $\boldsymbol{z}$ encodes the information of the entire source sequence $\boldsymbol{x}$, and is used to initialize the first hidden state of the RNN decoder.

Training Since we assume that target sequence $\boldsymbol{y}$ is a function of source sequence $\boldsymbol{x}$, we have $q_\phi(\boldsymbol{z}|\boldsymbol{y}) = q_\phi(\boldsymbol{z}|f(\boldsymbol{x})) \triangleq q_\phi(\boldsymbol{z}|\boldsymbol{x})$. Thus, the ELBo of the marginal log-likelihood is:

$$
\begin{aligned}
\log p_\theta(\boldsymbol{y}|\boldsymbol{x}) &\geq \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log \frac{p_\theta(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x})}{q_\phi(\boldsymbol{z}|\boldsymbol{x})}] \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{z})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{y}_1|\boldsymbol{y}_0, \boldsymbol{z}) + \sum_{t=2}^{T} \log p_\theta(\boldsymbol{y}_t|\boldsymbol{y}_{t-1})] - \mathrm{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) \\
&= \mathrm{ELBo}(\boldsymbol{y}, \boldsymbol{z}|\boldsymbol{x}; \phi, \theta)
\end{aligned}
\tag{3.1}
$$

where the variational posterior is a Gaussian distribution $q_\phi(\boldsymbol{z}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{z}|\mu_\phi(\boldsymbol{x}), diag(\sigma^2_\phi(\boldsymbol{x})))$ parameterized by the recognition model, and the prior is a standard Gaussian distribution $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}|\boldsymbol{0}, \boldsymbol{I})$. The first and second component of the ELBo are the expected reconstruction error of the generative model, and the last component is the KL divergence between the variational posterior and the prior. Maximizing the ELBo will push the KL divergence close to 0 and essentially force the variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ to match the prior $p(\boldsymbol{z})$. As prior work [20] demonstrates that the natural language text usually has multi-modal distribution in the latent space, the standard Gaussian prior constrains the variational posterior in representing complex latent variables for text. Therefore, choosing an appropriate $p(\boldsymbol{z})$ may help the variational posterior jump out of local optimal and learn a better approximation to the true posterior distribution $p(\boldsymbol{z}|\boldsymbol{x})$.

Inference The generative process of variational Seq2Seq models is different from the standard variational autoencoders [17, 18]. In order to encode the information from source sequence $\boldsymbol{x}$, the latent variable $\boldsymbol{z}$ is sampled from $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ instead of $p(\boldsymbol{z})$. When decoding the target sequence, I use greedy decoding [55] to generate predictions $\hat{\boldsymbol{y}}$.

Bad local optimal When directly optimizing the ELBo in Equation 3.1, I find that regardless of the source sequence $\boldsymbol{x}$, the decoder always predicts the same sentence on the

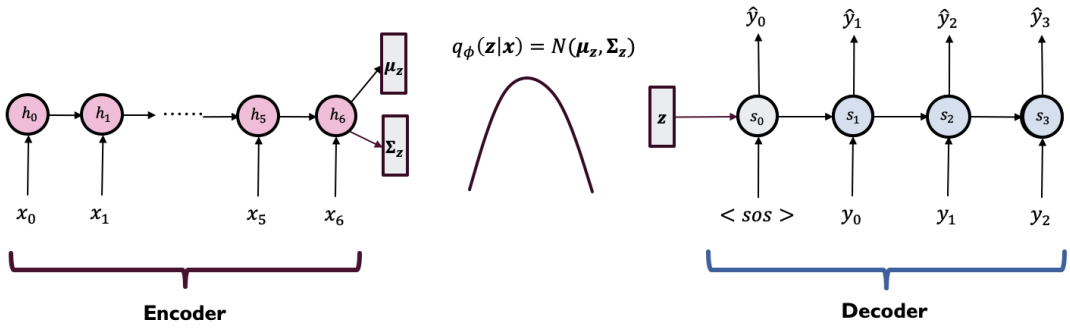

Figure 3.1: The architecture of variational Seq2Seq models.

test set. This issue indicates that the RNN decoder completely ignores the information from latent variable $\boldsymbol{z}$ and reduces to a RNN language model. By analyzing Equation 3.1, I find that the ELBo can be optimized by simply letting the variational posterior $q_\phi(\boldsymbol{z}|\boldsymbol{x})$ equal to the prior $p(\boldsymbol{z})$, and focusing on maximizing the reconstruction error of the decoder $\sum_{t=2}^{T} \log p_\theta(\boldsymbol{y}_t|\boldsymbol{y}_{t-1})$. As illustrated in Figure 3.1, the latent variable $\boldsymbol{z}$ is only used to initialize the initial decoder hidden state $\boldsymbol{s}_0$. The RNN decoder simply reduces to an auto-regressive model such that $p_\theta(\boldsymbol{y}_t|\boldsymbol{y}_{t-1})$ when $t >= 2$. In this case, the recognition model is trapped in a bad local optimal, and replacing the standard Gaussian prior with other more flexible priors can no longer help improve the performance, because the decoder does not rely on the latent variable $\boldsymbol{z}$ to generate target sequence $\boldsymbol{y}$.

3.2 Varitional Attention for Seq2Seq Models

Since the reason for being trapped in the bad local optimal is because the decoder ignores the source sequence information from latent variable $\boldsymbol{z}$, a legitimate solution is to introduce the information from source sequence at every decoding time step. Specifically, the attention mechanism [4] is an effective method which enables the decoder to incorporate information from source sequence at each decoding time step.

Attention Mechanism At each decoding time step t, the attention mechanism will compute an attention vector $\boldsymbol{c}_t$ (also called as context vector), which is a deterministic repre-

sentation of the dynamics between all encoder hidden states $\boldsymbol{h} = \{\boldsymbol{h}_1, \boldsymbol{h}_2, ..., \boldsymbol{h}_N\}$ and the previous decoder hidden state $\boldsymbol{s}_{t-1}$:

$$\boldsymbol{c}_t = \sum_{i=1}^{N} \alpha_{ti} \cdot \boldsymbol{h}_i \tag{3.2}$$

where the attention weight α_{ti} is computed as:

$$\alpha_{ti} = \frac{\exp(\boldsymbol{s}_{t-1} \cdot \mathbf{W}_a^T \cdot \boldsymbol{h}_i)}{\sum_{j=1}^{N} \exp(\boldsymbol{s}_{t-1} \cdot \mathbf{W}_a^T \cdot \boldsymbol{h}_j)} \tag{3.3}$$

where $\mathbf{W}_a$ is a weight matrix. A higher affinity of the previous decoder hidden state $\boldsymbol{s}_{t-1}$ with the ith encoder hidden state $\boldsymbol{h}_i$ will lead to a higher probability that the attention vector is drawn from the ith encoding position. The current decoder hidden state $\boldsymbol{s}_t$ is computed as:

$$\boldsymbol{s}_t = \text{RNN}(\boldsymbol{s}_{t-1}, \boldsymbol{y}_{t-1}, \boldsymbol{c}_t) \tag{3.4}$$

During training, $\boldsymbol{y}_{t-1}$ is the ground truth target word, while during testing, it becomes the predicted word sampled from the generative model $p_\theta(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \boldsymbol{c}_t) = \text{softmax}(\mathbf{W}_y \cdot \boldsymbol{s}_t)$, where $\mathbf{W}_y$ is a weight matrix.

Design of Variational Posteriors A related work on applying attention mechanism into the variational Seq2Seq framework is proposed by [50], where they treat both the last encoder hidden state $\boldsymbol{h}_N$ and the attention vector $\boldsymbol{c}_t$ as latent random variables. The general model architecture for this variational attention mechanism is demonstrated in Figure 3.2. More specifically, they model the variational posterior of attention vectors as a diagonal Gaussian distribution $q_\psi(\boldsymbol{c}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{c}|\mu_\psi(\boldsymbol{c}), diag(\sigma_\psi^2(\boldsymbol{c})))$. Note that at each decoding time step t, the variational Seq2Seq model will compute a corresponding attention vector $\boldsymbol{c}_t$, and pass it to the recognition model to obtain a different variational posterior $q_\psi(\boldsymbol{c}_t|\boldsymbol{x})$. They model the variational posterior of last encoder hidden state the same way as defined in section 3.1.

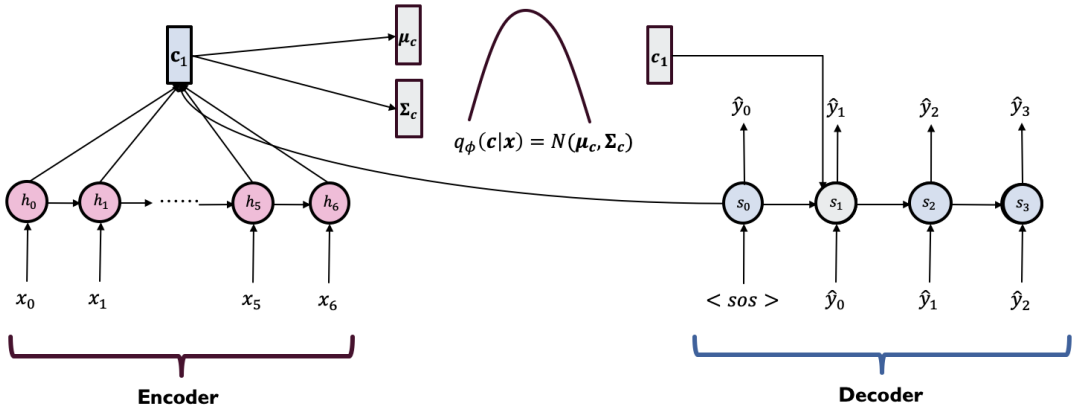

Figure 3.2: The architecture of Seq2Seq models with Variational Attention.

Design of Priors According to [50], they propose a sentence prior distributions for attention variable: $p(\boldsymbol{c}) = \mathcal{N}(\bar{\boldsymbol{h}}, \boldsymbol{I})$, which takes the average encoder hidden states as its mean. Since the attention vector by definition is a weighted sum of encoder hidden states (where the weight is computed as in Equation 3.3), introducing the average encoder hidden states as mean imposes a valid constraint on the latent attention variable, and can help variational posterior better approximate the true posterior distribution. The prior for the latent variable $\boldsymbol{z}$ follows the prior works [17, 18] and is defined as a standard Gaussian distribution: $p(\boldsymbol{z}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$.

ELBo Considering two latent variables $\boldsymbol{z}$ and $\boldsymbol{c}$, the ELBo in Equation 3.1 for variational Seq2Seq models now can be rewritten as:

$$\begin{aligned}
\text{ELBo}(\boldsymbol{y}, \boldsymbol{z}, \boldsymbol{c}|\boldsymbol{x}; \phi, \psi, \theta) &= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}), q_\psi(\boldsymbol{c}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{z}, \boldsymbol{c})] \\
&\quad - \text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - \text{KL}(q_\psi(\boldsymbol{c}|\boldsymbol{x})||p(\boldsymbol{c})) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}), q_\psi(\boldsymbol{c}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{y}_1|\boldsymbol{y}_0, \boldsymbol{z}, \boldsymbol{c}_1) + \sum_{t=2}^{T} \log p_\theta(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \boldsymbol{c}_t)] \\
&\quad - \text{KL}(q_\phi(\boldsymbol{z}|\boldsymbol{x})||p(\boldsymbol{z})) - \text{KL}(q_\psi(\boldsymbol{c}|\boldsymbol{x})||p(\boldsymbol{c}))
\end{aligned} \tag{3.5}$$

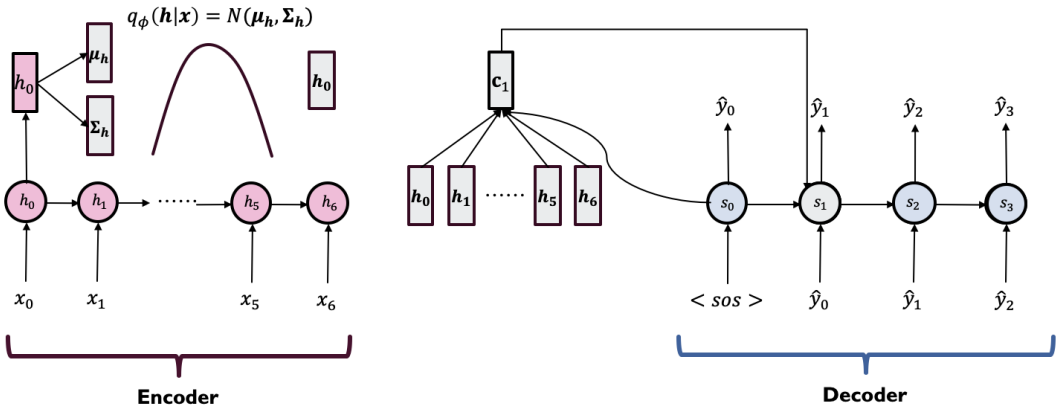

Figure 3.3: The architecture of Seq2Seq models with Variational Context.

where at each decoding time step t, the decoder makes prediction conditioning on both previous word $\boldsymbol{y}_{t-1}$ and attention variable $\boldsymbol{c}_t$, which enables information from source sequence to be incorporated into the decoder.

3.3 Variational Context for Seq2Seq Models

In this work, I explore a different way to utilize the information from source sequence. I propose to model encoder hidden states $\boldsymbol{h}$ as latent random variables, so that each word from source sequence $\boldsymbol{x}_n$ will have a variational posterior $q_\phi(\boldsymbol{h}_n|\boldsymbol{x}_n)$ parameterized by the recognition model, as shown in Figure 3.3. By replacing deterministic encoder hidden states with stochastic latent variables, the Seq2Seq model will obtain a larger search space for optimal context representations, which is helpful for avoiding the model being trapped in bad local optimal. Then, I apply the attention mechanism [4] to align decoder hidden state $\boldsymbol{s}_{t-1}$ with encoder latent variables $\boldsymbol{h}$ at each decoding time step t. To further help the Seq2Seq model jump out of the local optimal, I introduce some advanced priors learned from training data which may help the model to learn the complex data manifold in the latent space.

Design of Variational Posteriors Different from the prior work discussed in section 3.2, I treat encoder hidden states $\boldsymbol{h} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_N\}$ as latent random variables, and model the

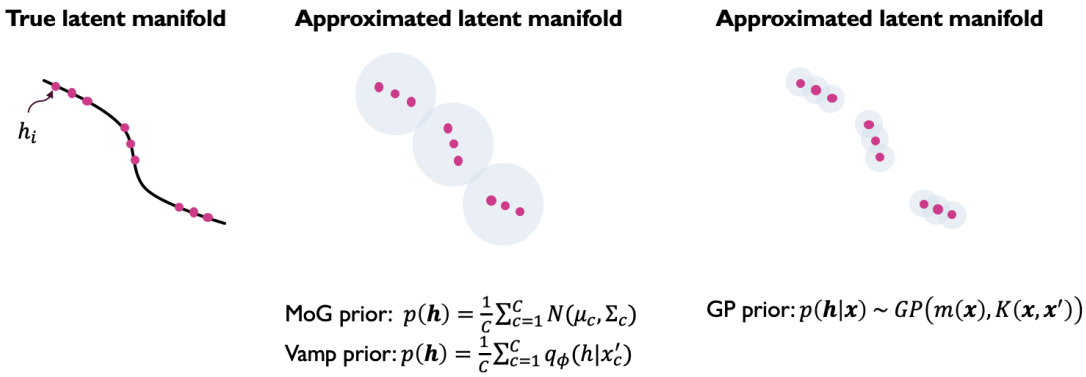

Figure 3.4: Comparison on approximated latent manifolds between different priors.

variational posterior as a diagonal Gaussian distribution $q_\phi(\boldsymbol{h}|\boldsymbol{x}) = \mathcal{N}(\boldsymbol{h}|\mu_\phi(\boldsymbol{x}), diag(\sigma^2_\phi(\boldsymbol{x})))$, where $\mu_\phi(\boldsymbol{x}_i) = f_\mu(\text{RNN}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}))$ and $\sigma^2_\phi(\boldsymbol{x}_i) = f_\sigma(\text{RNN}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1}))$. Note that the variational Seq2Seq model samples one latent variable $\boldsymbol{h}_i$ from $q_\phi(\boldsymbol{h}|\boldsymbol{x}_i)$ at each encoding position i, and then uses the sampled $\boldsymbol{h} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_N\}$ to compute the attention vector defined in Equation 3.2 and Equation 3.3 at each decoding time step, as shown in Figure 3.3.

Design of Priors In order to model the complex data manifold in the latent space and encourage the variational Seq2Seq model to find more optimal context representation, I propose to apply the following priors:

1. *Normal prior*: $p(\boldsymbol{h}) = \mathcal{N}(\boldsymbol{0}, \boldsymbol{I})$. This is a non-informative prior commonly used in variational autoencoders [17, 18].

2. *Mixture of Gaussians (MoG) prior*: $p(\boldsymbol{h}) = \frac{1}{C}\sum_{c=1}^{C} \mathcal{N}(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c)$, where C is the number of mixture components, $\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c$ are parameters learned during training, and $\boldsymbol{\Sigma}_c$ is a diagonal matrix. Each Gaussian component is assumed to cover different aspects of the source sequence.

3. *Mixture of Variational Posteriors (Vamp) prior* [22]: $p(\boldsymbol{h}) = \frac{1}{C}\sum_{c=1}^{C} q_\phi(\boldsymbol{h}|\boldsymbol{x}'_c)$, where C is the number of pseudo inputs, $\boldsymbol{x}'_c$ is a pseudo input which only has a unique out-of-vocabulary token, and the embedding of the unique out-of-vocabulary token is learned during training. Following the prior work [22], I learn a multi-modal

prior via pseudo-inputs, and assume each component represents a different semantic or syntactic meaning of the source sequence.

4. *Gaussian Process (GP) prior*: $p(\boldsymbol{h}|\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}'))$, where the mean function $m(\boldsymbol{x})$ is the encoder RNN such that $m(\boldsymbol{x}_i) = \text{RNN}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1})$ at each encoding position i, and the kernel function $K(\boldsymbol{x}, \boldsymbol{x}')$ is defined as:

$$K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} \mathbf{0} & , \boldsymbol{x}_i \neq \boldsymbol{x}_j \\ \boldsymbol{\Sigma}_\lambda & , \boldsymbol{x}_i = \boldsymbol{x}_j \end{cases}$$

such that the source word $\boldsymbol{x}_i$ at different encoding position is independent with each other. The intuition of the design of the mean and kernel function is that it can introduce some randomness from the data variance around each hidden state. I prove the correctness of the proposed GP prior in the next paragraph. The difference between the proposed GP prior and other priors is that the proposed GP prior is an ad-hoc prior which only applies to the current encoded hidden state $\boldsymbol{h}_i$.

An intuitive visualization of the approximated latent manifold for different priors is illustrated in Figure 3.4. The GoM prior and Vamp prior will lead the model to approximate the latent manifold within a group of clusters, while the ad-hoc GP prior will lead the model to approximate the latent manifold around each individual data point.

Proof of Gaussian Process (GP) Prior Let $\mathcal{D} = \{(\boldsymbol{x}_1, \boldsymbol{h}_1), ..., (\boldsymbol{x}_{i-1}, \boldsymbol{h}_{i-1})\}$ be the observed training set with Gaussian random noise, where $\boldsymbol{h}_i = f(\boldsymbol{x}_i) + \epsilon$ and $\epsilon \sim \mathcal{N}(\mathbf{0}, \boldsymbol{\Sigma}_\lambda)$. $\boldsymbol{x}_i$ is the current source word, we want to predict the function of outputs $\boldsymbol{h}_i$. Let $\boldsymbol{h} = \{\boldsymbol{h}_1, ..., \boldsymbol{h}_{i-1}\}$, we have the joint distribution:

$$\begin{pmatrix} \boldsymbol{h} \\ \boldsymbol{h}_i \end{pmatrix} \sim \mathcal{N}\left(\begin{pmatrix} m(\boldsymbol{x}) \\ m(\boldsymbol{x}_i) \end{pmatrix}, \begin{pmatrix} K(\boldsymbol{x}, \boldsymbol{x}) + \boldsymbol{\Sigma}_\lambda & , K(\boldsymbol{x}, \boldsymbol{x}_i) \\ K(\boldsymbol{x}_i, \boldsymbol{x}) & , K(\boldsymbol{x}_i, \boldsymbol{x}_i) \end{pmatrix} \right) \tag{3.6}$$

The posterior distribution of output $\boldsymbol{h}_i$ becomes:

$$
\begin{aligned}
p(\boldsymbol{h}_i|\boldsymbol{x}_i, \boldsymbol{x}, \boldsymbol{h}) &= \mathcal{N}(\boldsymbol{\mu}_i, \boldsymbol{\Sigma}_i) \\
\boldsymbol{\mu}_i &= m(\boldsymbol{x}_i) + K(\boldsymbol{x}_i, \boldsymbol{x}) \cdot (K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \boldsymbol{\Sigma}_\lambda)^{-1} \cdot (f - m(\boldsymbol{x})) \\
\boldsymbol{\Sigma}_i &= K(\boldsymbol{x}_i, \boldsymbol{x}_i) - K(\boldsymbol{x}_i, \boldsymbol{x}) \cdot (K(\boldsymbol{x}_i, \boldsymbol{x}_i) + \boldsymbol{\Sigma}_\lambda)^{-1} \cdot K(\boldsymbol{x}, \boldsymbol{x}_i)
\end{aligned} \tag{3.7}
$$

In this work, I assume that each input data point $\boldsymbol{x}_i$ is independent of each other, and define kernel function as:

$$
K(\boldsymbol{x}_i, \boldsymbol{x}_j) = \begin{cases} \mathbf{0} & , \boldsymbol{x}_i \neq \boldsymbol{x}_j \\ \boldsymbol{\Sigma}_\lambda & , \boldsymbol{x}_i = \boldsymbol{x}_j \end{cases}
$$

where $\boldsymbol{\Sigma}_\lambda$ is the data variance when sampling at encoding position i. So we have the posterior distribution of the ith encoder hidden state $p(\boldsymbol{h}_i|\boldsymbol{x}_i, \boldsymbol{x}, \boldsymbol{h}) = \mathcal{N}(m(\boldsymbol{x}_i), \boldsymbol{\Sigma}_\lambda)$, where the mean function is defined to be the encoder RNN, i.e., $m(\boldsymbol{x}_i) = \text{RNN}(\boldsymbol{x}_i, \boldsymbol{h}_{i-1})$.

Since each encoder hidden state can be sampled from the above posterior distribution, we claim all encoder hidden states can be sampled from the Gaussian Process $p(\boldsymbol{h}|\boldsymbol{x}) \sim GP(m(\boldsymbol{x}), K(\boldsymbol{x}, \boldsymbol{x}'))$.

ELBo Since I treat encoder hidden states as latent variables, the ELBo in Equation 3.1 of the marginal log-likelihood is rewritten as:

$$
\begin{aligned}
\text{ELBo}(\boldsymbol{y}, \boldsymbol{h}|\boldsymbol{x}; \phi, \theta) &= \mathbb{E}_{q_\phi(\boldsymbol{h}|\boldsymbol{x})}[\log p_\theta(\boldsymbol{y}|\boldsymbol{h})] - \text{KL}(q_\phi(\boldsymbol{h}|\boldsymbol{x})||p(\boldsymbol{h})) \\
&= \mathbb{E}_{q_\phi(\boldsymbol{h}|\boldsymbol{x})}[\sum_{t=1}^{T} \log p_\theta(\boldsymbol{y}_t|\boldsymbol{y}_{t-1}, \boldsymbol{c}_t)] - \text{KL}(q_\phi(\boldsymbol{h}|\boldsymbol{x})||p(\boldsymbol{h}))
\end{aligned} \tag{3.8}
$$

where the attention vector $\boldsymbol{c}_t$ is computed based on latent variable $\boldsymbol{h}$ sampled from the variational posterior $q_\phi(\boldsymbol{h}|\boldsymbol{x})$ as well as the previous decoder hidden state $\boldsymbol{s}_{t-1}$ as defined in Equation 3.2 and Equation 3.3, $\boldsymbol{y}_0$ is a special start of sequence token SOS.

3.4 Geometric Analysis on latent representations

The motivation of conducting geometric analysis on latent manifold is to analyze whether variational autoencoders learn a good approximation to the true low-dimensional latent manifold of text data. Since the true latent manifold cannot be obtained in practice, I use the latent manifold in the standard Seq2Seq model as a strong baseline to the true latent manifold. If variational Seq2Seq models also learn a good approximation to the true latent manifold, I expect their latent manifold to share similar geometric property with the latent manifold in the standard Seq2Seq model.

I use local Principal Component Analysis (PCA) [56] to explain the geometric property of latent manifolds. In geometric data analysis, local PCA is shown to be a useful tool to capture intrinsic dimensions of data manifolds in high-dimensional space. Given a set of K-dimensional latent representations $\boldsymbol{Z} = \{\boldsymbol{z}^{(i)}\}_{i=1}^{M}$, PCA applies a orthogonal linear transformation to transform $\boldsymbol{z}$ into a new coordinate system [57], such that the greatest variance of the data comes to lie on the first coordinate, the second greatest variance lies on the second coordinate, and so on. By computing and comparing the data variance ratio in each coordinate, we can get a sense of how the data distributed in the high-dimensional latent space.

In this work, I follow the prior work [56] to conduct local PCA on latent representations sampled from the recognition model. The algorithm can be conclued in three major steps:

Step 1. Obtain latent representations from recognition model: For the standard Seq2Seq model, directly compute all latent representations $\{\boldsymbol{h}^{(i)}\}_{i=1}^{M}$ from the encoder, for all $\boldsymbol{x}^{(i)}$ in training set. For variational Seq2Seq models, sample their corresponding latent representations from the mean of their variational posterior: $\boldsymbol{z}^{(i)} \sim q_\phi(\boldsymbol{z}|\boldsymbol{x}^{(i)})$, $\boldsymbol{c}^{(i)} \sim q_\psi(\boldsymbol{c}|\boldsymbol{x}^{(i)})$ or $\boldsymbol{h}^{(i)} \sim q_\phi(\boldsymbol{h}|\boldsymbol{x}^{(i)})$, for all $\boldsymbol{x}^{(i)}$ in training set.

Step 2. Randomly select a local centroid and find its closest neighbours: After obtaining all latent representations from the training set, randomly select a latent representation $\boldsymbol{h}^{(i)}$ as the local centroid $\bar{\boldsymbol{h}}$. Then, compute the Euclidean distance between the local centroid $\bar{\boldsymbol{h}}$ and other remaining latent representations $\{\boldsymbol{h}^{(i)}\}_{i=1}^{M-1}$:

$$L2 = ||\bar{\boldsymbol{h}} - \boldsymbol{h}^{(i)}||_2^2, \qquad i = 1, ..., M-1 \tag{3.9}$$

I sort the latent representations in terms of the Euclidean distance in a descending order, and select the top K number of latent representations $\boldsymbol{H}_{nn} = \{\boldsymbol{h}^{(j)}\}_{j=1}^{K}$, where K is a hyper-parameter and $K << M$, as the nearest neighbours of the local centroid $\bar{\boldsymbol{h}}$.

Step 3. Conduct PCA on the local centroid and its closest neighbors: After having the nearest neighbours of the local centroid $\bar{\boldsymbol{h}}$, subtract them by $\bar{\boldsymbol{h}}$:

$$\bar{\boldsymbol{H}}_{nn} = \{\boldsymbol{h}^{(j)} - \bar{\boldsymbol{h}}\}_{j=1}^{K} \tag{3.10}$$

Then, compute its covariance matrix, and do eigenvalue decomposition on it:

$$\boldsymbol{C} = \frac{1}{M}\bar{\boldsymbol{H}}_{nn}\bar{\boldsymbol{H}}_{nn}^T \tag{3.11}$$

$$\boldsymbol{C} = \boldsymbol{W}\boldsymbol{\Lambda}\boldsymbol{W}^T \tag{3.12}$$

where $\boldsymbol{\Lambda} = \{\lambda_k\}_{k=1}^{K}$ is the diagonal matrix of eigenvalues of $\bar{\boldsymbol{H}}_{nn}\bar{\boldsymbol{H}}_{nn}^T$. I use the eigenvalue λ_k in $\boldsymbol{\Lambda}$ to explain the data variance in the k-th latent dimension: the higher the eigenvalue λ_k is, the larger data variance the k-th latent dimension will have. More specifically, I compute the explained variance ratio of each eigen vector:

$$r_k^2 = \frac{\lambda_k^2}{\sum_{j=1}^{K}\lambda_j^2} \tag{3.13}$$

I sort the explained variance ratio in a descending order, and preserve the eigen vectors which summed up to 95% cumulative explained variance ratio as the principal components.

Do step 2 and step 3 iteratively until we sample enough local data points to identify the data distribution in each latent dimension.

3.5 Empirical Experiments

3.5.1 Paraphrase Generation

I evaluate our models in the paraphrase generation task. The goal of paraphrase generation is to generate a paraphrase of a given sentence, which preserves the maximum semantic meaning of the source sentence but expressed in a different way.

Dataset and evaluation metrics I evaluate our models on the Quora Question Pair Dataset [1], and the Twitter URL Paraphrasing Dataset [58] [2]. Both datasets contain positive and negative examples of paraphrases, and I only keep the positive examples for our experiments as in prior work of paraphrase generation [13, 59]. For the Quora dataset, I follows the configuration of [13] and split the data into 100K training pairs, 30K testing pairs and 3K validation pairs. For the Twitter dataset, I divided it into 110K training pairs, 3K testing pairs and 1K validation pairs.

For the evaluation metrics, I follow prior works [4, 9, 13] and use BLEU-1 and BLEU-2 score [45] to evaluate the quality of generated sentences compared with ground truth target sentences [3]. In order to measure the diversity of the generated sentences, I follow prior works [50, 60] and use distinct metrics. The distinct metrics compute the percentage of distinct unigrams (the *Dist-1*) or bigrams (the *Dist-2*) in generated sentences respectively.

[1] `https://www.kaggle.com/c/quora-question-pairs`

[2] `https://languagenet.github.io`

[3] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

Competitive models In order to ensure that the performance improvement comes from the proposed priors, I do not apply state-of-the-art paraphrase generation models but instead implement a simple RNN encoder-decoder with attention mechanism (Seq2Seq) as proposed in [4]. I compare our models with the baseline Seq2Seq and the Seq2Seq model with variational attention (VA-Seq2Seq) proposed in [50].

I first implement the Seq2Seq model with variational context using a standard normal prior (VC-Seq2Seq + Normal). Following [22], I implement the VC-Seq2Seq model with MoG prior (VC-Seq2Seq + MoG) and Vamp prior (VC-Seq2Seq + Vamp) as discussed in section 3.3. Finally, I implement the VC-Seq2Seq model with Gaussian Process prior (VC-Seq2Seq + GP).

Experimental setup For the baseline RNN encoder-decoder with attention mechanism model (Seq2Seq), I use two single-layer GRU as the encoder and decoder. The input is a sequence of tokens whose maximum length is set to be 20. A word embedding layer with dimension 300 is applied to get the continuous representation of the sentence. The hidden state dimension for both encoder and decoder is 300. I use Adam optimizer with learning rate 5×10^{-5} for training. I train all models with 200 epochs, do auto-validation on the validation set after every training epoch, and save the model with the lowest validation loss. The VA-Seq2Seq and VC-Seq2Seq models share the same configuration with the baseline Seq2Seq model, and add additional prior and posterior with the dimension size of 100.

Result analysis Table 3.1 and Table 3.2 show the performance of different models on the Quora test set and Twitter test set respectively. I observe that the Seq2Seq model is a strong baseline in terms of BLEU scores, and other proposed models can hardly outperform it. However, the VC-Seq2Seq with GP prior manages to achieve comparable performance with the Seq2Seq model, and generates sentence with higher diversity than all other models. The observations are consistent in both datasets, which shows that GP prior is helpful for both learning informative context representations of source sequences and generating

Model	Prior	BLEU-1	BLEU-2	Dist-1	Dist-2
Prior work					
1. Seq2Seq	/	**48.3**	**24.0**	0.0140	0.0992
2. VA-Seq2Seq	$\mathcal{N}\left(\bar{\boldsymbol{h}}, \boldsymbol{I}\right)$	43.5	20.5	0.0153	0.1061
Our work					
3. VC-Seq2Seq + Normal	$\mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$	38.9	17.3	0.0134	0.0855
4. VC-Seq2Seq + MoG	$\frac{1}{C}\sum_{c=1}^{C}\mathcal{N}\left(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$	38.2	16.8	0.0151	0.1146
5. VC-Seq2Seq + Vamp	$\frac{1}{C}\sum_{c=1}^{C} q_\phi(\boldsymbol{z}\|\boldsymbol{x}'_c)$	39.6	17.7	0.0131	0.0843
6. VC-Seq2Seq + GP	$\prod_{i=0}^{N}\mathcal{N}(\mu_\lambda(x_i), \boldsymbol{\Sigma}_\lambda)$	47.5	23.2	**0.0155**	**0.1154**

Table 3.1: Paraphrase generation performance on Quora test set.

Model	Prior	BLEU-1	BLEU-2	Dist-1	Dist-2
Prior work					
1. Seq2Seq	/	**39.7**	**24.2**	0.0858	0.3161
2. VA-Seq2Seq	$\mathcal{N}\left(\bar{\boldsymbol{h}}, \boldsymbol{I}\right)$	36.9	21.0	0.0850	0.3391
Our work					
3. VC-Seq2Seq + Normal	$\mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$	27.5	13.4	0.0671	0.2602
4. VC-Seq2Seq + MoG	$\frac{1}{C}\sum_{c=1}^{C}\mathcal{N}\left(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$	29.3	13.9	0.0717	0.3070
5. VC-Seq2Seq + Vamp	$\frac{1}{C}\sum_{c=1}^{C} q_\phi(\boldsymbol{z}\|\boldsymbol{x}'_c)$	32.2	16.6	0.0830	0.3515
6. VC-Seq2Seq + GP	$\prod_{i=0}^{N}\mathcal{N}(\mu_\lambda(x_i), \boldsymbol{\Sigma}_\lambda)$	39.1	23.7	**0.1004**	**0.4284**

Table 3.2: Paraphrase generation performance on Twitter test set.

Source	hacking a password ?
Target	how do i hack a password ?
Predictions	
Seq2Seq	what are some of the best ways to improve your communication skills ?
VA-Seq2Seq	what is the best way to find someone who is a good question ?
VC-Seq2Seq + Normal	how do i find out online with a little UNK of number - number in a week ?
VC-Seq2Seq + MoG	how can i hack somebody 's whatsapp account if you do not remember the email or a password ?
VC-Seq2Seq + Vamp	how can i hack my instagram account ?
VC-Seq2Seq + GP	how do i hack a twitter account ?

Table 3.3: Sample paraphrase generation results on Quora test set.

Source	hackers used new weapons in attack on u.s. internet
Target	hackers used new weapons to disrupt major websites across u.s. and they only attacked the united states
Predictions	
Seq2Seq	hackers used new weapons to disrupt major websites across u.s.
VA-Seq2Seq	hackers used new weapons to double major websites across u.s.
VC-Seq2Seq + Normal	new york times # us # hackers used new weapons to double major websites across u.s.
VC-Seq2Seq + MoG	hackers used new weapons to disrupt major websites across u.s. , officials say
VC-Seq2Seq + Vamp	hackers used new weapons to disrupt major websites across u.s. the number of these are the biggest time .
VC-Seq2Seq + GP	hackers used new weapons to disrupt major websites across u.s. # ddos attack # iot @cnn

Table 3.4: Sample paraphrase generation results on Twitter test set.

	Source: hacking a password ? **Target**: how do i hack a password ?	
	VC-Seq2Seq + Normal	**VC-Seq2Seq + GP**
1	*how do i find out online with a little UNK of number - number in a week ?*	*how do i hack a twitter account ?*
2	i have any good password and i have tried zoosk and i have tried zoosk and UNK them from UNK	your best way to hack a password ?
3	and how do i get rid of them ?	my way to hack ?
4	and i have saved back or not ?	my way to get a good password ?
5	how long should i spend on top of number ?	my UNK is not added as a password ?
6	i do not have a time work and i can get them to my friend from your own search ?	my way to control my twitter ?

Table 3.5: Different $\boldsymbol{h}$ sampled from variation posterior $q_\phi(\boldsymbol{h}|\boldsymbol{x})$ on Quora test set.

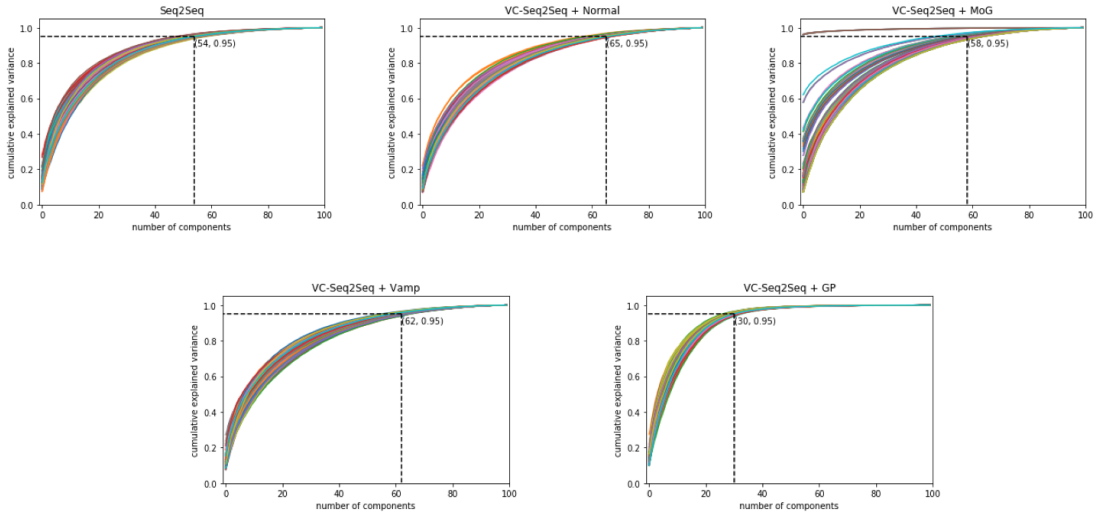

Figure 3.5: Cumulative explained variance plots of latent representations from Seq2Seq and VC-Seq2Seq with different priors on Quora dataset.

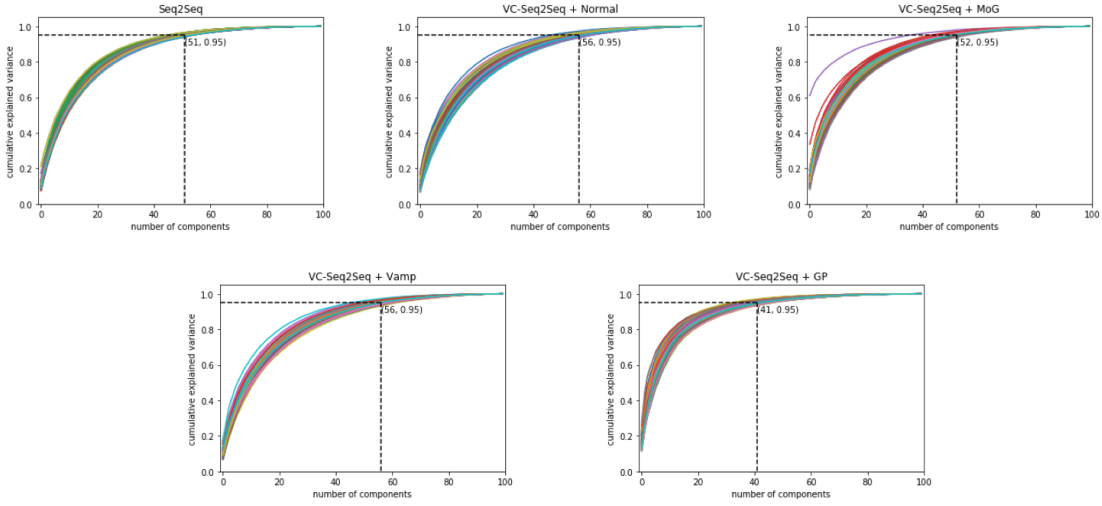

Figure 3.6: Cumulative explained variance plots of latent representations from Seq2Seq and VC-Seq2Seq with different priors on Twitter dataset.

diversified target sequences.

For the evaluation of generating sequence that matches target sequence, I compare the BLEU-1 and BLEU-2 scores across different models. Table 3.1 and Table 3.2 show that the VC-Seq2Seq with GP prior can generate high quality target sentences compared with other variational Seq2Seq models. Table 3.3 and Table 3.4 provide some generated sequences from different models. I observe that the VC-Seq2Seq with GP prior is able to generate paraphrases which have different expressions and also preserve the semantic meaning of source sentences.

For the evaluation of generating diversified sequences, I compare the *Dist-1* and *Dist-2* across different models. Table 3.1 and Table 3.2 demonstrate that the VC-Seq2Seq with GP prior generates most diversified sentences, which suggests that the GP prior is helpful for the model to search optimal context representations in the latent space.

For the evaluation of encoding informative context representations, I sample different latent representations from the variational posterior of different models to see if the model can generate sentences that preserve similar semantic meanings. Table 3.5 shows the prediction results of sampling different latent variable $\boldsymbol{h}$ from the variational posterior $q_\phi(\boldsymbol{h}|\boldsymbol{x})$, where the italic sentence at line 1 is sampled from the mean of the variational posterior. Comparing the generated sentences between normal prior and GP prior, I find that the GP prior is able to introduce various expressions which are still semantically close to the source sentence, while the standard normal prior fails to introduce relevant expressions.

For the evaluation of learning a good approximation to the true latent manifold, I conduct local PCA on latent representations from Seq2Seq and VC-Seq2Seq with different priors, as shown in Figure 3.5 and Figure 3.6. As discussed in section 3.4, if varational Seq2Seq models learn a good approximation to the true latent manifold, it will have a low-dimensional latent representation in the latent space. From Figure 3.5 and Figure 3.6, I find that VC-Seq2Seq with GP prior has the smallest total number of principal components, which not only outperforms other variational Seq2Seq models, but also outperforms the

strong baseline Seq2Seq model. This suggests that introducing GP prior is beneficial for the variational Seq2Seq models to approximating the true latent manifold.

3.5.2 Style Transfer

I also evaluate our models in the style transfer task, more specifically, the formality transfer [49] task. The goal of formality transfer is to translate an informal sentence into a formal sentence, which preserves the maximum semantic meaning of the source informal sentence.

Dataset and evaluation metrics I evaluate our models on the Grammarly's Yahoo Answers Formality Corpus (GYAFC) [49]. The GYAFC dataset has two sub-domains: Entertainment & Music (E&M), Family & Relationships (F&R). For the Entertainment & Music (E&M) domain, it has 52,595 training pairs, 2,877 validation pairs and 1,416 testing pairs. For the Family & Relationships (F&R) domain, it has 51,967 training pairs, 2,788 validation pairs and 1,332 testing pairs.

For the evaluation metrics, I use BLEU-1 and BLEU-2 score [45] to evaluate the quality of generated sentences compared with ground truth target sentences [4], and use distinct metrics [50, 60] to measure the diversity of the generated sentences. The distinct metrics compute the percentage of distinct unigrams (the *Dist-1*) or bigrams (the *Dist-2*) in generated sentences respectively.

Experimental setup The experimental setup and competitive models are the same with the paraphrase generation task, please refer to subsection 3.5.1 for details.

Result analysis Table 3.6 and Table 3.7 show the performance of different priors in the E&M test set and F&R test set respectively. Similar to the observations in paraphrase generation task, the Seq2Seq model is a strong baseline in terms of BLEU scores, and other

[4] `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`

Model	Prior	BLEU-1	BLEU-2	Dist-1	Dist-2
Prior work					
1. Seq2Seq	/	**37.4**	**12.2**	0.0400	0.2398
2. VA-Seq2Seq	$\mathcal{N}\left(\bar{\boldsymbol{h}}, \boldsymbol{I}\right)$	36.2	11.2	0.0320	0.1948
Our work					
3. VC-Seq2Seq + Normal	$\mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$	33.3	9.5	0.0324	0.1816
4. VC-Seq2Seq + MoG	$\frac{1}{C}\sum_{c=1}^{C}\mathcal{N}\left(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$	30.5	8.3	**0.0707**	**0.3302**
5. VC-Seq2Seq + Vamp	$\frac{1}{C}\sum_{c=1}^{C} q_\phi(\boldsymbol{z}\|\boldsymbol{x}'_c)$	29.0	7.4	0.0233	0.1228
6. VC-Seq2Seq + GP	$\prod_{i=0}^{N}\mathcal{N}(\mu_\lambda(x_i), \boldsymbol{\Sigma}_\lambda)$	37.2	12.1	0.0402	0.2360

Table 3.6: Formality transfer performance on GYAFC (E&M) test set.

Model	Prior	BLEU-1	BLEU-2	Dist-1	Dist-2
Prior work					
1. Seq2Seq	/	41.7	15.8	0.0274	0.2140
2. VA-Seq2Seq	$\mathcal{N}\left(\bar{\boldsymbol{h}}, \boldsymbol{I}\right)$	38.1	13.4	0.0277	0.2027
Our work					
3. VC-Seq2Seq + Normal	$\mathcal{N}\left(\mathbf{0}, \boldsymbol{I}\right)$	37.7	12.9	0.0277	0.1836
4. VC-Seq2Seq + MoG	$\frac{1}{C}\sum_{c=1}^{C}\mathcal{N}\left(\boldsymbol{\mu}_c, \boldsymbol{\Sigma}_c\right)$	37.5	12.2	**0.0602**	**0.3783**
5. VC-Seq2Seq + Vamp	$\frac{1}{C}\sum_{c=1}^{C} q_\phi(\boldsymbol{z}\|\boldsymbol{x}'_c)$	36.4	11.8	0.0280	0.1867
6. VC-Seq2Seq + GP	$\prod_{i=0}^{N}\mathcal{N}(\mu_\lambda(x_i), \boldsymbol{\Sigma}_\lambda)$	**42.3**	**16.1**	0.0335	0.2384

Table 3.7: Formality transfer performance on GYAFC (F&R) test set.

Source	just a dum funny question hahahaha
Target	just a senseless , funny question .
Predictions	
Seq2Seq	just a question , it is just a funny question .
VA-Seq2Seq	just a question about this question .
VC-Seq2Seq + Normal	i think that you were a little bit funny .
VC-Seq2Seq + MoG	it is a very long time . it seems as though it is a very funny movie .
VC-Seq2Seq + Vamp	a man can not get a UNK .
VC-Seq2Seq + GP	it is a good question , but it is funny .

Table 3.8: Sample formality transfer results on GYAFC (E&M) test set.

Source	dont forget to choose and vote for best answer ! ! !
Target	do not forget to choose and vote for the best answer !
Predictions	
Seq2Seq	do not give up the best for the best and i wish to find a good friend to answer !
VA-Seq2Seq	do not forget to be happy and tell the truth .
VC-Seq2Seq + Normal	do not forget to be happy with the best answer .
VC-Seq2Seq + MoG	do not forget to choose and vote for the best answer .
VC-Seq2Seq + Vamp	do not forget to the best answer , and be patient and have to be able to find the answer
VC-Seq2Seq + GP	do not forget to answer and vote for the best answer .

Table 3.9: Sample formality transfer results on GYAFC (F&R) test set.

	Source: just a dum funny question hahahaha **Target**: just a senseless , funny question .	
	VC-Seq2Seq + Normal	**VC-Seq2Seq + GP**
1	*i think that you were a little bit funny .*	*it was a good question , but it is funny .*
2	you * thinking of a funny question .	it is a good question .
3	i have a friend who is a bad day .	it is very funny .
4	was a good question .	it is very good .
5	is a great time .	it is not a good thing .
6	is more of the time .	a good thing .

Table 3.10: Different $\boldsymbol{h}$ sampled from variation posterior $q_\phi(\boldsymbol{h}|\boldsymbol{x})$ on GYAFC (E&M) test set.

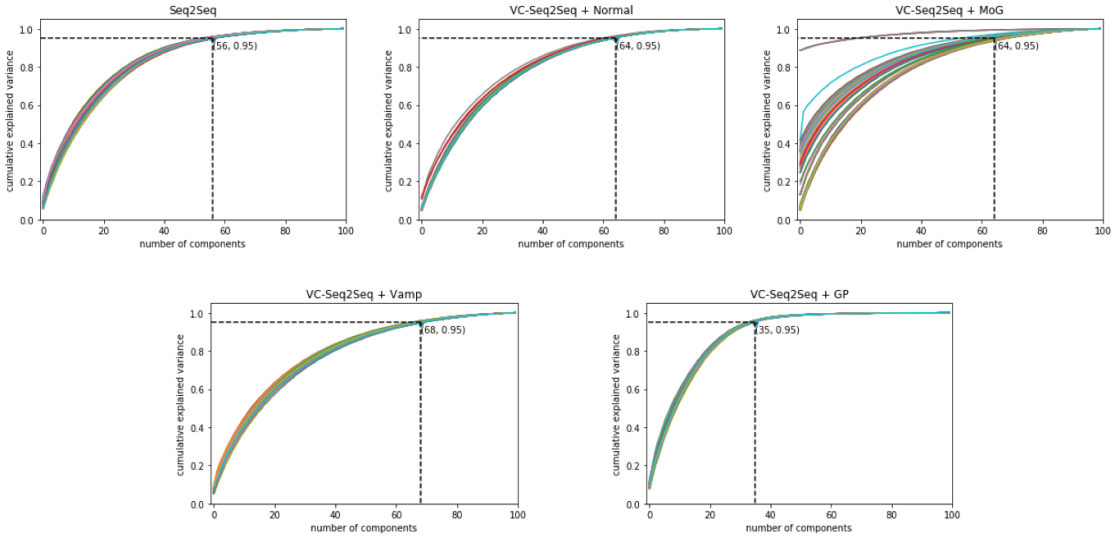

Figure 3.7: Cumulative explained variance plots of latent representations from Seq2Seq and VC-Seq2Seq with different priors on GYAFC (E&M) dataset.

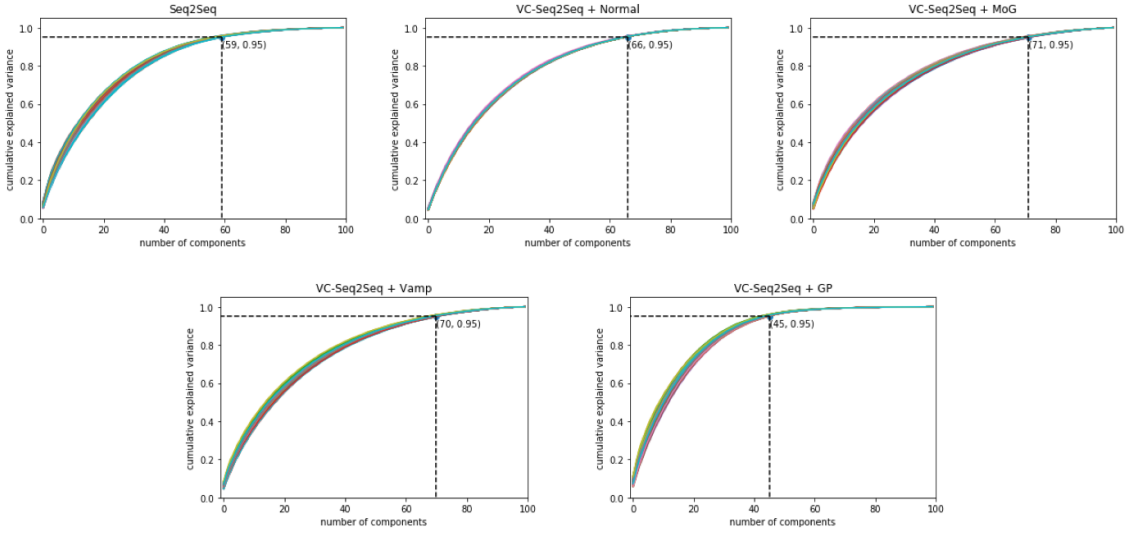

Figure 3.8: Cumulative explained variance plots of latent representations from Seq2Seq and VC-Seq2Seq with different priors on GYAFC (F&R) dataset.

proposed priors can hardly outperform it. Besides, the VC-Seq2Seq with MoG prior generates most diversified texts, but does not have good performance in matching the ground truth target sentences. The VC-Seq2Seq with GP prior manages to achieve comparable performance with the Seq2Seq model in terms of BLEU scores, and generate more diversified texts than the Seq2Seq model. On both datasets, the VC-Seq2Seq with GP prior outperforms the VA-Seq2Seq model not only in matching ground truth target sequences, but also in generating diversified texts.

For the evaluation of generating sequence that matches target sequence, I compare the BLEU-1 and BLEU-2 scores across different models. Table 3.6 and Table 3.7 show that the VC-Seq2Seq with GP prior achieves comparable results with the strong baseline Seq2Seq model on E&M test set, and even outperforms the Seq2Seq model on F&R test set. Table 3.9 and Table 3.8 provide some generated sentences sampled from the test set, which demonstrate that the VC-Seq2Seq with GP prior is able to translate informal sentence to formal sentence, and also preserve the semantic meaning.

For the evaluation of generating diversified sequences, I compare the *Dist-1* and *Dist-2* across different models. Table 3.6 and Table 3.7 show that the VC-Seq2Seq with MoG prior generates sentences with the highest diversity, but it has poor performance in matching ground truth target sentences. While the VC-Seq2Seq with GP prior generates sentences with the second highest diversity, and also good performance in matching ground truth target sentences.

For the evaluation of encoding informative context representations, I sample different latent representations from the variational posterior of different models. Table 3.10 shows the prediction results of sampling different latent variable $\boldsymbol{h}$ from the variational posterior $q_\phi(\boldsymbol{h}|\boldsymbol{x})$. The observation is consistent with the paraphrase generation task: the GP prior is able to introduce various expressions which are still semantically close to the source sentence, while the standard normal prior fails to introduce relevant expressions.

For the evaluation of learning a good approximation to the true latent manifold, I con-

duct local PCA on latent representations from Seq2Seq and VC-Seq2Seq with different priors, as shown in Figure 3.7 and Figure 3.8. In formality transfer task, the VC-Seq2Seq with GP also manages to learn the most compact and low-dimensional representations in the latent space, which validates our assumption that introducing appropriate prior can help variational Seq2Seq models learn better approximation to the true latent manifold.

3.6 Contributions

In this chapter, I analyze the reason for the poor performance of adapting the variational autoencoders framework to Seq2Seq models, and introduce the variational attention Seq2Seq framework to solve the problem. To help the variational attention Seq2Seq models better approximate the true latent distribution, I propose to replace the standard normal prior with a GP prior. The GP prior applies the encoder RNN as its mean function, and simplifies the kernel function to an identity matrix, which imposes randomness around each latent variable without considering the correlation between different latent variables. Experiments on paraphrase generation and formality transfer tasks show that the proposed variational Seq2Seq model with GP prior achieves comparable results with the standard Seq2Seq model in generating sentences that match the ground truth target sentences, and the generated sentences have larger diversity than the standard Seq2Seq model. I also conduct local PCA to analyze the local geometry of latent representations from the proposed variational Seq2Seq model and the standard Seq2Seq model. The results show that the proposed variational Seq2Seq model with GP prior can learn a better low-dimensional latent representations than the standard Seq2Seq model, which validates our assumption that learning an appropriate prior can help the variational Seq2Seq model avoid bad local optimal and better approximate the true latent distribution.

CHAPTER 4
AN EMPIRICAL COMPARISON ON LEARNING ALGORITHMS FOR SEQ2SEQ MODELS

Work described in this chapter was undertaken in collaboration with Yangfeng Ji, and published at EMNLP 2019 [61].

Generating text from given context involves decoding words step by step from a large vocabulary. To learn a decoder, supervised learning which maximizes the likelihood of tokens often suffers from the exposure bias [29]. Although both reinforcement learning and imitation learning have been widely used to alleviate the bias, the lack of direct comparison leads to only a partial image on their benefits. In this chapter, I present an empirical study on how reinforcement learning and imitation learning can help alleviate the exposure bias. I evaluate all learning algorithms on paraphrase generation task, since generating paraphrases is a fundamental research problem that could benefit many other downstream NLP applications, such as machine translation [4], document summarization [62], question ansIring [5], etc.

The rest of this chapter is organized as follows: section 4.1 explains the exposure bias problem in training recurrent neural networks with maximum likelihood estimation (also called *teacher forcing*), and discusses some popular learning algorithms to alleviate the exposure bias; section 4.2 introduces a unified learning framework to incorporate different learning algorithms to better understand how they can help alleviate the exposure bias; section 4.3 defines evaluation metrics and demonstrates empirical experiment results for different learning algorithms; section 4.4 concludes the benefits and limitations of different learning algorithms.

4.1 Exposure Bias in Teacher Forcing

Recurrent neural networks [63] are a typical choice for generative models to generate texts. During training, the model predicts the next word conditioning on previous ground truth words. However, at test time, since ground truth words are not available, the model has to predict the next word conditioning on its previous generated words. This training strategy is called *teacher forcing*, which trains the model using words sampled from data distribution, but tests the model using words sampled from model distribution. If the model distribution is very different from the data distribution, the generative model will propagate and accumulate prediction errors at test time. [29] defines this phenomena as the exposure bias problem.

To address this problem, prior works [29, 30, 31] suggest to utilize the exploration strategy in reinforcement learning (RL). A typical way of using RL in practice is to use model predictions at training time and optimize the final evaluation metrics instead of maximizing the likelihood of the ground truth words. However, training with the RL algorithms is not trivial and often hardly works in practice [32]. In the middle ground between RL and *teacher forcing*, a well-known category is imitation learning (IL) [33, 34], which randomly introduces model predictions during training with a *schedule rate* and obtains the optimal policy by maximizing the likelihood of ground truth words.

4.2 A Unified Learning Framework

To better understand the benefits and limitations of different learning algorithms, I propose a unified learning framework to include some popular learning algorithms as special cases, such as the REINFORCE algorithm [35] in RL and the DAGGER algorithm [34] in IL. I further propose several variant learning algorithms based on the unified learning framework to have a complete comparison between all possible learning algorithms in RL and IL.

I build up the learning framework on paraphrase generation task. Given an input sen-

tence $\boldsymbol{x} = (x_1, x_2, \cdots, x_S)$ with length S, a paraphrase generation model outputs a new sentence $\boldsymbol{y} = (y_1, y_2, \cdots, y_T)$ with length T that shares the same meaning with $\boldsymbol{x}$. The widely adopted framework on paraphrase generation is the encoder-decoder framework [64]. The encoder reads sentence $\boldsymbol{x}$ and represents it as a single numeric vector or a set of numeric vectors. The decoder defines a probability function $p(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta})$, where $\boldsymbol{y}_{\leq t-1} = (y_1, y_2, \ldots, y_{t-1})$ and $\boldsymbol{\theta}$ is the collection of model parameters,

$$p(y_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta}) = \text{softmax}(\mathbf{W}\boldsymbol{h}_t) \tag{4.1}$$

with $\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, y_{t-1}, \boldsymbol{x})$, where $\boldsymbol{f}$ as a nonlinear transition function and $\mathbf{W} \in \boldsymbol{\theta}$ as a parameter matrix. I use the *pointer-generator model* [1] as the base model, which is state-of-the-art model on paragraph generation [13]. I skip the detail explanation of this model and please refer to [1] for further information.

4.2.1 Basic Learning Algorithms

To facilitate the comparison between RL and IL, I propose a unified framework with the following objective function. Given a training example $(\boldsymbol{x}, \boldsymbol{y})$, the objective function is defined as

$$L(\boldsymbol{\theta}) = \big\{ \sum_{t=1}^{T} \log \pi_\theta(\tilde{y}_t \mid \boldsymbol{h}_t) \big\} \cdot r(\tilde{\boldsymbol{y}}, \boldsymbol{y}), \tag{4.2}$$

Following the terminology in RL and IL, I rename $P(\tilde{y}_t \mid \boldsymbol{y}_{\leq t-1}, \boldsymbol{x}; \boldsymbol{\theta})$ as the the policy function $\pi_\theta(\tilde{y}_t \mid \boldsymbol{h}_t)$. That implies taking an action based on the current observation, where the action is *picking* a word $\tilde{y}_t$ from the vocabulary $\mathcal{V}$. $r(\tilde{\boldsymbol{y}}, \boldsymbol{y})$ is a reward function with $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$ if $\tilde{\boldsymbol{y}} = \boldsymbol{y}$. In our experiments, I use the ROUGE-2 score [65] as the reward function. Algorithm 1 presents how to optimize $L(\boldsymbol{\theta})$ in the online learning fashion. As shown in the pseudocode, the **schedule rates** (α, β) and the **decoding function** `Decode`$(\cdot)$ are the keys to understand the special cases of this unified framework.

Algorithm 1 Online learning algorithm

Input: A training example $(\boldsymbol{x}^{(i)}, \boldsymbol{y}^{(i)})$, current schedule rates $\alpha^{(i)}, \beta^{(i)} \in [0, 1]$, learning rate η
Initialize $L(\boldsymbol{\theta}) \leftarrow 0$
for $t = 1, \ldots, T$ **do**
  $p_1, p_2 \sim \text{Uniform}(0, 1)$
  $\tilde{y}_{t-1} \leftarrow y_{t-1}$ **if** $(p_1 < \alpha^{(i)})$ **else** $\hat{y}_{t-1}$
  $\boldsymbol{h}_t = \boldsymbol{f}(\boldsymbol{h}_{t-1}, \tilde{y}_{t-1}, \boldsymbol{x})$
  $\hat{y}_t \leftarrow \texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$
  $\tilde{y}_t \leftarrow y_t$ **if** $(p_2 < \beta^{(i)})$ **else** $\hat{y}_t$
  $L(\boldsymbol{\theta}) \leftarrow L(\boldsymbol{\theta}) + \log \pi(\tilde{y}_t \mid \boldsymbol{h}_t)$
end for
$\delta\boldsymbol{\theta} \leftarrow \nabla_{\boldsymbol{\theta}} L(\boldsymbol{\theta}) \cdot r(\tilde{\boldsymbol{y}}^{(i)}, \boldsymbol{y}^{(i)})$
$\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \eta \cdot \delta\boldsymbol{\theta}$

The REINFORCE Algorithm. When $\alpha = 0$, $\beta = 0$, and $\texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$ is defined as as:

$$\texttt{Decode}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})) = \text{Random\_Sampling}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})), \tag{4.3}$$

Specifically, when $\alpha = \beta = 0$, both $\tilde{y}_{t-1}$ and $\tilde{y}_t$ will choose the sampled values from the Decode function with policy π_θ. It essentially samples a trajectory from the decoder $(\hat{y}_1, \hat{y}_2, \ldots, \hat{y}_T)$ as in the REINFORCE algorithm. The reward is $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = r(\hat{\boldsymbol{y}}, \boldsymbol{y})$ once it has the entire trajectory $\hat{\boldsymbol{y}}$.

The DAGGER Algorithm. When $0 < \alpha < 1$, $\beta = 1$, and $\texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$ is defined as as:

$$\texttt{Decode}(\pi_\theta(y \mid \boldsymbol{h}_{t-1})) = \text{Random\_Sampling}(\pi_\theta(y' \mid \boldsymbol{h}_{t-1})). \tag{4.4}$$

Depending the value of α, $\tilde{y}_{t-1}$ will choose between the ground truth y_{t-1} and decoded value $\hat{y}_{t-1}$ with the function defined in Equation 4.4. On the other hand, $\tilde{y}_t$ will always choose the ground truth y_t as $\beta = 1$. Since $\tilde{\boldsymbol{y}} = \boldsymbol{y}$, we have $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$ and the reward can be ignored from Equation 4.2. In imitation learning, ground truth sequence $\boldsymbol{y}$ is called expert actions. The DAGGER algorithm [34] is also called scheduled sampling [15] in

recent deep learning literature. To be accurate, α is dynamically changed during training. Typically, it starts from 1 and gradually decays to a certain value along with iterations. As shown in our experiments, the selection of decay scheme has a big impact on model performance.

The MLE Algorithm. Besides, there is a trivial case when $\alpha = 1, \beta = 1$. In this case, $\tilde{y}_{t-1}$ and $\tilde{y}_t$ are equal to y_{t-1} and y_t respectively, and $r(\tilde{\boldsymbol{y}}, \boldsymbol{y}) = 1$. Optimizing the objective function in Equation 4.2 is reduced to the maximum likelihood estimation (MLE).

4.2.2 Other Variant Algorithms

Inspired by the three special cases mentioned above, I offer other algorithm variants with different combinations of (α, β), while the decoding function $\texttt{Decode}(\pi(y \mid \boldsymbol{h}_t))$ in the same as Equation 4.3 in all following variants.

- **REINFORCE-GTI** (REINFORCE with Ground Truth Input): $\alpha = 1$, $\beta = 0$. Unlike the REINFORCE algorithm, REINFORCE-GTI restricts the input to the decoder can only be ground truth words, which means $\tilde{y}_{t-1} = y_{t-1}$. This is a popular implementation in the deep reinforcement learning for Seq2Seq models [66].

- **REINFORCE-SO** (REINFORCE with Sampled Output): $\alpha = 1$, $0 < \beta < 1$. In terms of choosing the value of $\tilde{y}_t$ as output from the decoder, REINFORCE-SO allows $\tilde{y}_t$ to select the ground truth y_t with probability β.

- **REINFORCE-SIO** (REINFORCE with Sampled Input and Output): $0 < \alpha < 1$, $0 < \beta < 1$. Instead of always taking the ground truth y_{t-1} as input, REINFORCE-SIO further relaxes the constraint in REINFORCE-SO and allows $\tilde{y}_{t-1}$ to be the decoded value $\hat{y}_{t-1}$ with probability α.

Unless specified explicitly, an additional requirement when $0 < \alpha, \beta < 1$ is that its value decays to a certain value during training, which by default is 0.

4.3 Empirical Experiments

Dataset and Evaluation Metrics. I evaluate our models on the Quora Question Pair Dataset [1], and the Twitter URL Paraphrasing Dataset [58] [2]. Both datasets contain positive and negative examples of paraphrases, and I only keep the positive examples for our experiments as in prior work of paraphrase generation [13, 59]. For the Quora dataset, I follows the configuration of [13] and split the data into 100K training pairs, 30K testing pairs and 3K validation pairs. For the Twitter dataset, since our model cannot deal with the negative examples as [13] do, I just obtain the 1-year 2,869,657 candidate pairs from `https://languagenet.github.io`, and filter out all negative examples. Finally, I divided the remaining dataset into 110K training pairs, 3K testing pairs and 1K validation pairs.

To align with prior works [4, 9, 13], I use the following evaluation metrics to compare our models with other state-of-art neural networks: ROUGE-1 and ROUGE-2 [65], BLEU with up to bi-grams [45]. For the convenience of comparison, I also calculate the average of the scores.

Competitive Systems. I compare our models with four competitive systems on paraphrase generation: the sequence-to-sequence model [4, Seq2seq], the Reinforced by Matching framework [13, RbM], the Residual LSTM [9, Res-LSTM], and the Discriminator LSTM model [59, Dis-LSTM]. Among these competitive systems, the RbM [13] is more closely related to our work, since we both use the pointer-generator as the base model and apply some reinforcement learning algorithms for policy learning.

Experimental Setup. For all experiments, our model has 256-dimensional hidden states and 128-dimensional word embeddings. Since the pointer-generator model has the ability to deal with the OOV words, I choose a small vocabulary size of 5k, and we train the word

[1] `https://www.kaggle.com/c/quora-question-pairs`
[2] `https://languagenet.github.io`

embedding from scratch. I also truncate both the input and output sentences to 20 tokens.

For the training part, I first pre-train a pointer generator model using MLE, and then fine-tune this model with the REINFORCE, DAGGER and other variant learning algorithms respectively. In the **pre-training** phase, I use the Adagrad optimizer with learning rate 0.15 and an initial accumulator value of 0.1; use gradient clipping with a maximum gradient norm of 2; and do auto-evaluation on the validation set every 1000 iterations, in order to save the best model with the lowest validation loss. In the **fine-tuning** phase, I use the Adam optimizer with learning rate 10^{-5}; use gradient clipping with the same setting in pre-training; and do auto-evaluation on the validation set every 10 iterations.

When applying the REINFORCE and its variant algorithms, I compute the reward as follows:

$$r(\tilde{\boldsymbol{y}}_n, \boldsymbol{y}) = \text{ROUGE-2}(\tilde{\boldsymbol{y}}_n, \boldsymbol{y}) - \frac{1}{N}\sum_{n=1}^{N}\text{ROUGE-2}(\tilde{\boldsymbol{y}}_n, \boldsymbol{y}) \tag{4.5}$$

where N is the total number of sentences generated by random sampling, in all the experiments, we set $N = 4$.

At test time, I use beam search with beam size 8 to generate the paraphrase sentence.

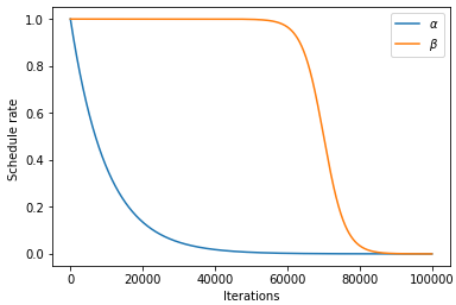

Figure 4.1: The schedule sampling rate for α and β

According to [15], I define the schedule rate $\alpha^{(i)} = k^i$ (where $0 < k < 1$, i is the ith training iteration), and $\beta^{(i)} = k/(k+exp(i/k))$ (where $k > 1$, i is the ith training iteration). In the experiments shown in 4.1, for the schedule rate α, I set $k = 0.9999$; for the schedule

	Models	Schedule Rate		Evaluation Metrics			
		α	β	ROUGE-1	ROUGE-2	BLEU	Avg
1	Seq2Seq	-	-	58.77	31.47	36.55	42.26
2	Res-LSTM	-	-	59.21	32.43	37.38	43.00
3	RbM	-	-	64.39	38.11	43.54	48.68
4	Dis-LSTM	-	-	-	44.90	45.70	45.30
5	Pre-trained MLE	$\alpha = 1$	$\beta = 1$	66.72	47.70	54.01	56.14
6	Reinforce	$\alpha = 0$	$\beta = 0$	67.00	47.91	54.06	56.32
7	Reinforce-GTI	$\alpha = 1$	$\beta = 0$	67.03	48.10	54.23	56.45
8	Reinforce-SO	$\alpha = 1$	$\beta \to 0$	66.88	47.95	54.16	56.33
9	Reinforce-SIO	$\alpha \to 0$	$\beta \to 0$	67.62	48.99	55.19	57.26
10	Dagger	$\alpha \to 0$	$\beta = 1$	67.64	48.96	55.06	57.22
11	Dagger*	$\alpha = 0.5$	$\beta = 1$	**68.34**	**49.99**	**55.75**	**58.02**

Table 4.1: Paraphrase generation performance on Quora test set. The results of competitive systems are reprinted from prior work: line 1 – 3 are obtained from [13], line 4 is obtained from [59].

	Models	Schedule Rate		Evaluation Metrics			
		α	β	ROUGE-1	ROUGE-2	BLEU	Avg
1	Pre-trained MLE	$\alpha = 1$	$\beta = 1$	58.49	43.84	38.45	46.92
2	Reinforce	$\alpha = 0$	$\beta = 0$	58.67	44.06	38.46	47.06
3	Reinforce-GTI	$\alpha = 1$	$\beta = 0$	58.58	43.89	38.42	46.96
4	Reinforce-SO	$\alpha = 1$	$\beta \to 0$	58.58	43.89	38.41	46.96
5	Reinforce-SIO	$\alpha \to 0$	$\beta \to 0$	58.82	44.10	38.85	47.25
6	Dagger	$\alpha \to 0$	$\beta = 1$	58.84	44.24	38.95	47.34
7	Dagger*	$\alpha = 0.2$	$\beta = 1$	**58.95**	**44.34**	**39.04**	**47.44**

Table 4.2: Paraphrase generation performance on Twitter test set. Since the dataset I obtained is different from [13], I do not directly compare the results with the prior works.

rate β, I set $k = 3000$. The schedule rate curve is shown in Figure 4.1.

Result Analysis. Table 4.1 shows the model performances on the Quora test set, and Table 4.2 shows the model performances on the Twitter test set. For the Quora dataset, all our models outperform the competitive systems with a large margin. I suspect the reason is because I ran the development set during training on-the-fly, which is not the experimental setup used in [13].

For both datasets, DAGGER with a fixed (α, β) gives the best performance among all the algorithm variants. The difference between DAGGER and DAGGER* is that, in DAGGER, I use the decay function on α at each iteration, $\alpha \leftarrow k \cdot \alpha$ with $k = 0.9999$. In our experiments, I also try different decaying rates, and present the best results I obtained. The selection of α depends on the specific task: for the Quora dataset, I find $\alpha = 0.5$ gives us the optimal policy; for the Twitter dataset, I find $\alpha = 0.2$ gives us the optimal policy.

The reinforcement learning and imitation learning algorithms can help the encoder-decoder model alleviate the exposure bias. Additional training with whichever variant algorithms can certainly enhance the generation performance over the pre-trained model, as shown in line 6 – 11 from Table 4.1 and line 2 – 7 from Table 4.2. This observation is consistent with many previous works of using RL/IL in NLP.

However, the improvement of the REINFORCE algorithm is very small, only 0.18 on the average score on Quora dataset, and 0.14 on the average score on Twitter dataset, which indicates that the model cannot jump out of the local optimal. Among all RL-based algorithms, REINFORCE-SIO algorithm performs best, which suggests introducing some ground truth information can help the model cannot jump out of the local optimal, as shown in line 9 from Table 4.1 and line 5 from Table 4.2.

I also observe that the average improvement in the Twitter dataset is not as significant as in the Quora dataset. Since in the Twitter dataset, one source sentence shares several different paraphrases, while in the Quora dataset, one source sentence only corresponds to one paraphrase. For the pointer-generator model, the Twitter dataset is more challenging than the Quora dataset.

Overall, in this particular setting of paraphrase generation, I found that DAGGER is much easier to use than the REINFORCE algorithm, as it optimizes its policy by maximizing the likelihood of ground truth words (expert actions). Although, picking a good decay function α can be really tricky. On the other hand, the REINFORCE algorithm (together with its variants) could only outperform the pre-trained baseline with a small margin.

4.4 Contributions

In this chapter, I perform an empirical study on some reinforcement learning and imitation learning algorithms for paraphrase generation. The empirical experiment results show that both reinforcement learning and imitation learning algorithms are helpful in alleviating the exposure bias. By comparing the performance between the REINFORCE algorithm and the REINFORCE-SIO algorithm, I find that standard reinforcement learning is likely to trap in the local optimal, and introducing some ground truth information can help improve the performance. By comparing the performance between the DAGGER algorithm and other algorithms, I find that DAGGER is the most efficient learning algorithms in paraphrase generation, but the choice of the schedule rate is task-specific. Fine-tuning the pointer-generator model with DAGGER can outperform the state-of-the-art method with a large margin.

CHAPTER 5
CONCLUSION

With the goal of generating accurate and diversified texts, I explore different approaches to learn an optimal Seq2Seq model.

The first approach is to apply the variational autoencoders framework on the Seq2Seq model to learn a probabilistic distribution of latent representations. I adopt the attention mechanism into the variational Seq2Seq framework, and propose to learn a GP prior to help the model learn informative and diversified latent context representations. Experiments in paraphrase generation and style transfer show that the proposed method is able to consistently generate accurate and diversified texts.

The second approach is to try different learning algorithms to alleviate the training-inference discrepancy when training Seq2Seq models. I present empirical comparison between different popular learning algorithms (e.g. maximum likelihood estimation, REINFORCE, DAGGER) in an unified framework, and analyze their benefits and limitations in alleviating the training-inference discrepancy. Experiments in paraphrase generation show that DAGGER is most efficient learning algorithm, but the choice of its schedule rate is task-specific. Fine-tuning state-of-the-art model with DAGGER algorithm can further achieve a large margin performance improvement.

The Seq2Seq framework is a widely used framework for natural language generation. My study in improving the generation performance of Seq2Seq models can be applied in a wide range of sequence-to-sequence generation applications, such as dialogue system, question answering, machine translation, etc. While current proposed methods focus on sentence-level text generation, future work could be conducted on paragraph-level or document-level text generation, where the input text becomes more complicated and challenging.

REFERENCES

[1] A. See, P. J. Liu, and C. D. Manning, "Get to the point: Summarization with pointer-generator networks," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 1073–1083.

[2] I. Androutsopoulos and P. Malakasiotis, "A survey of paraphrasing and textual entailment methods," *Journal of Artificial Intelligence Research*, vol. 38, pp. 135–187, 2010.

[3] I. V. Serban, A. Sordoni, R. Lowe, L. Charlin, J. Pineau, A. Courville, and Y. Bengio, "A hierarchical latent variable encoder-decoder model for generating dialogues," in *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

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

[5] B. McCann, N. S. Keskar, C. Xiong, and R. Socher, "The natural language decathlon: Multitask learning as question answering," *arXiv preprint arXiv:1806.08730*, 2018.

[6] S. Zhao, X. Lan, T. Liu, and S. Li, "Application-driven statistical paraphrase generation," in *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2 - Volume 2*, ser. ACL '09, Suntec, Singapore: Association for Computational Linguistics, 2009, pp. 834–842, ISBN: 978-1-932432-46-6.

[7] C. Quirk, C. Brockett, and W. Dolan, "Monolingual machine translation for paraphrase generation," in *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004, pp. 142–149.

[8] R. Barzilay and L. Lee, "Learning to paraphrase: An unsupervised approach using multiple-sequence alignment," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, ser. NAACL '03, Edmonton, Canada: Association for Computational Linguistics, 2003, pp. 16–23.

[9] A. Prakash, S. A. Hasan, K. Lee, V. Datla, A. Qadir, J. Liu, and O. Farri, "Neural paraphrase generation with stacked residual lstm networks," in *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, 2016, pp. 2923–2934.

[10] K. Cho, A. Courville, and Y. Bengio, "Describing multimedia content using attention-based encoder-decoder networks," *IEEE Transactions on Multimedia*, vol. 17, no. 11, pp. 1875–1886, 2015.

[11] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural computation*, vol. 1, no. 2, pp. 270–280, 1989.

[12] T. Shen, T. Lei, R. Barzilay, and T. Jaakkola, "Style transfer from non-parallel text by cross-alignment," in *Advances in neural information processing systems*, 2017, pp. 6830–6841.

[13] Z. Li, X. Jiang, L. Shang, and H. Li, "Paraphrase generation with deep reinforcement learning," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

[14] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. MIT press, 2018.

[15] S. Bengio, O. Vinyals, N. Jaitly, and N. Shazeer, "Scheduled sampling for sequence prediction with recurrent neural networks," in *Advances in Neural Information Processing Systems*, 2015, pp. 1171–1179.

[16] D. M. Blei, M. I. Jordan, *et al.*, "Variational inference for dirichlet process mixtures," *Bayesian analysis*, vol. 1, no. 1, pp. 121–143, 2006.

[17] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[18] D. J. Rezende, S. Mohamed, and D. Wierstra, "Stochastic backpropagation and approximate inference in deep generative models," in *International Conference on Machine Learning*, 2014, pp. 1278–1286.

[19] D. Rezende and S. Mohamed, "Variational inference with normalizing flows," in *International Conference on Machine Learning*, 2015, pp. 1530–1538.

[20] I. V. Serban, A. Ororbia II, J. Pineau, and A. Courville, "Piecewise latent variables for neural variational text processing," in *Proceedings of the 2nd Workshop on Structured Prediction for Natural Language Processing*, 2017, pp. 52–62.

[21] J. Xu and G. Durrett, "Spherical latent spaces for stable variational autoencoders," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4503–4513.

[22] J. Tomczak and M. Welling, "Vae with a vampprior," in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 1214–1223.

[23] H. Xu, W. Chen, J. Lai, Z. Li, Y. Zhao, and D. Pei, "On the necessity and effectiveness of learning the prior of variational auto-encoder," *arXiv preprint arXiv:1905.13452*, 2019.

[24] Y. Gal, M. Van Der Wilk, and C. E. Rasmussen, "Distributed variational inference in sparse gaussian process regression and latent variable models," in *Advances in neural information processing systems*, 2014, pp. 3257–3265.

[25] J. Hensman, N. Fusi, and N. D. Lawrence, "Gaussian processes for big data," in *Uncertainty in Artificial Intelligence*, Citeseer, 2013, p. 282.

[26] M. Bauer, M. van der Wilk, and C. E. Rasmussen, "Understanding probabilistic sparse gaussian process approximations," in *Advances in neural information processing systems*, 2016, pp. 1533–1541.

[27] F. P. Casale, A. Dalca, L. Saglietti, J. Listgarten, and N. Fusi, "Gaussian process prior variational autoencoders," in *Advances in Neural Information Processing Systems*, 2018, pp. 10 369–10 380.

[28] A. M. Lamb, A. G. A. P. Goyal, Y. Zhang, S. Zhang, A. C. Courville, and Y. Bengio, "Professor forcing: A new algorithm for training recurrent networks," in *Advances In Neural Information Processing Systems*, 2016, pp. 4601–4609.

[29] M. Ranzato, S. Chopra, M. Auli, and W. Zaremba, "Sequence level training with recurrent neural networks," *arXiv preprint arXiv:1511.06732*, 2015.

[30] S. Shen, Y. Cheng, Z. He, W. He, H. Wu, M. Sun, and Y. Liu, "Minimum risk training for neural machine translation," in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2016, pp. 1683–1692.

[31] D. Bahdanau, P. Brakel, K. Xu, A. Goyal, R. Lowe, J. Pineau, A. Courville, and Y. Bengio, "An actor-critic algorithm for sequence prediction," *arXiv preprint arXiv:1607.07086*, 2016.

[32] P. Dayan and Y. Niv, "Reinforcement learning: The good, the bad and the ugly," *Current opinion in neurobiology*, vol. 18, no. 2, pp. 185–196, 2008.

[33] H. Daumé, J. Langford, and D. Marcu, "Search-based structured prediction," *Machine learning*, vol. 75, no. 3, pp. 297–325, 2009.

[34] S. Ross, G. Gordon, and D. Bagnell, "A reduction of imitation learning and structured prediction to no-regret online learning," in *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 2011, pp. 627–635.

[35] R. J. Williams, "Simple statistical gradient-following algorithms for connectionist reinforcement learning," *Machine learning*, vol. 8, no. 3-4, pp. 229–256, 1992.

[36] A. Mnih and K. Gregor, "Neural variational inference and learning in belief networks," in *Proceedings of the 31st International Conference on International Conference on Machine Learning-Volume 32*, 2014, pp. II–1791.

[37] M. I. Jordan, Z. Ghahramani, T. S. Jaakkola, and L. K. Saul, "An introduction to variational methods for graphical models," *Machine learning*, vol. 37, no. 2, pp. 183–233, 1999.

[38] M. J. Wainwright and M. I. Jordan, "Introduction to variational methods for graphical models," *Foundations and Trends in Machine Learning*, vol. 1, pp. 1–103, 2008.

[39] G. E. Box and G. C. Tiao, *Bayesian inference in statistical analysis*. John Wiley & Sons, 2011, vol. 40.

[40] C. Cremer, X. Li, and D. Duvenaud, "Inference suboptimality in variational autoencoders," in *International Conference on Machine Learning*, 2018, pp. 1078–1086.

[41] D. P. Kingma, T. Salimans, R. Jozefowicz, X. Chen, I. Sutskever, and M. Welling, "Improved variational inference with inverse autoregressive flow," in *Advances in neural information processing systems*, 2016, pp. 4743–4751.

[42] R. Krishnan, D. Liang, and M. Hoffman, "On the challenges of learning with inference networks on sparse, high-dimensional data," in *International Conference on Artificial Intelligence and Statistics*, 2018, pp. 143–151.

[43] Y. Kim, S. Wiseman, A. Miller, D. Sontag, and A. Rush, "Semi-amortized variational autoencoders," in *International Conference on Machine Learning*, 2018, pp. 2678–2687.

[44] C.-W. Huang, A. Touati, L. Dinh, M. Drozdzal, M. Havaei, L. Charlin, and A. Courville, "Learnable explicit density for continuous latent space and variational inference," *arXiv preprint arXiv:1710.02248*, 2017.

[45] K. Papineni, S. Roukos, T. Ward, and W.-J. Zhu, "Bleu: A method for automatic evaluation of machine translation," in *Proceedings of the 40th annual meeting on association for computational linguistics*, Association for Computational Linguistics, 2002, pp. 311–318.

[46] A. Lavie and M. J. Denkowski, "The meteor metric for automatic evaluation of machine translation," *Machine translation*, vol. 23, no. 2-3, pp. 105–115, 2009.

[47] H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, and M. Marchand, "Domain-adversarial neural networks," *arXiv preprint arXiv:1412.4446*, 2014.

[48] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, and V. Lempitsky, "Domain-adversarial training of neural networks," *The Journal of Machine Learning Research*, vol. 17, no. 1, pp. 2096–2030, 2016.

[49] S. Rao and J. Tetreault, "Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018, pp. 129–140.

[50] H. Bahuleyan, L. Mou, O. Vechtomova, and P. Poupart, "Variational attention for sequence-to-sequence models," in *Proceedings of the 27th International Conference on Computational Linguistics*, 2018, pp. 1672–1682.

[51] K. Tolias, I. Kourouklides, and S. Chatzis, "Amortized context vector inference for sequence-to-sequence networks," *arXiv preprint arXiv:1805.09039*, 2018.

[52] B. Zhang, D. Xiong, J. Su, H. Duan, and M. Zhang, "Variational neural machine translation," in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, 2016, pp. 521–530.

[53] C. Zhou and G. Neubig, "Morphological inflection generation with multi-space variational encoder-decoders," in *Proceedings of the CoNLL SIGMORPHON 2017 Shared Task: Universal Morphological Reinflection*, 2017, pp. 58–65.

[54] ——, "Multi-space variational encoder-decoders for semi-supervised labeled sequence transduction," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, 2017, pp. 310–320.

[55] U. Germann, "Greedy decoding for statistical machine translation in almost linear time," in *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, Association for Computational Linguistics, 2003, pp. 1–8.

[56] N. Kambhatla and T. K. Leen, "Dimension reduction by local principal component analysis," *Neural computation*, vol. 9, no. 7, pp. 1493–1516, 1997.

[57] I. T. Jolliffe, "Mathematical and statistical properties of population principal components," *Principal component analysis*, pp. 10–28, 2002.

[58] W. Lan, S. Qiu, H. He, and W. Xu, "A continuously growing dataset of sentential paraphrases," in *Proceedings of the 2017 Conference on Empirical Methods in Nat-*

ural Language Processing, Copenhagen, Denmark: Association for Computational Linguistics, Sep. 2017, pp. 1224–1234.

[59] B. N. Patro, V. K. Kurmi, S. Kumar, and V. P. Namboodiri, "Learning semantic sentence embeddings using pair-wise discriminator," *arXiv preprint arXiv:1806.00807*, 2018.

[60] J. Li, M. Galley, C. Brockett, J. Gao, and B. Dolan, "A diversity-promoting objective function for neural conversation models," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 110–119.

[61] W. Du and Y. Ji, "An empirical comparison on imitation learning and reinforcement learning for paraphrase generation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, 2019, pp. 6014–6020.

[62] S. Chopra, M. Auli, and A. M. Rush, "Abstractive sentence summarization with attentive recurrent neural networks," in *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2016, pp. 93–98.

[63] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh annual conference of the international speech communication association*, 2010.

[64] K. Cho, B. van Merriënboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, "Learning phrase representations using rnn encoder–decoder for statistical machine translation," in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1724–1734.

[65] C.-Y. Lin, "ROUGE: A package for automatic evaluation of summaries," in *Text Summarization Branches Out*, Barcelona, Spain: Association for Computational Linguistics, Jul. 2004, pp. 74–81.

[66] Y. Keneshloo, T. Shi, N. Ramakrishnan, and C. K. Reddy, "Deep reinforcement learning for sequence-to-sequence models," *IEEE Transactions on Neural Networks and Learning Systems*, 2019.