Chapter 2

A Generative Approach for Paraphrase Generation

We present in this chapter a deep generative framework that learns the two joint distributions of source and target language spaces with another hidden space. The model not only manages to reproduce paraphrases of a sentence by sampling through its distribution, but is also capable of generating a pair of parallel sentences from one vector in the hidden space.

2.1 Related Work

Deep Generative Models, deep neural networks that try to learn underlying data distribution of training data and generate new data with some variations, have recently become a hot topic in deep learning for its high and diverse applicability. Among the existing generative models, Generative Adversarial Network (GAN) (Goodfellow et al. (2014)) and Variational Auto-Encoder (VAE) (Kingma & Welling (2014)) have gained a lot of attention from public.

For the paraphrase generation task, deep learning-based approaches have been introduced recently and achieved great success. The first major work that uses deep frameworks to generate paraphrase includes Prakash et al. (2016), where the authors propose a stacked recurrent neural network with residual connections. Later, Gupta et al. (2018) introduces a deep generative framework based on a VAE seq-to-seq model (Bowman et al. (2016)), they condition the decoder on the input sentence during the inference to force the model to produce a paraphrase of the input. Alternatively, Li et al. (2018) uses a deep reinforcement learning architecture with a generator and an evaluator that can judge whether two sentences are paraphrases. Most recently, Roy & Grangier (2019) proposes an unsupervised model that can generate paraphrases using only monolingual data. Their model rely on Vector Quantised-Variational AutoEncoder (VQ-VAE) (van den Oord et al. (2017)), a model representing discrete latent variables rather than continuous variables as in VAE.

Our work is similar to Gupta et al. (2018) in that we also combine NMT with VAE to generate new data. The difference is that we employ 2 VAE frameworks in source and target sides to learn the joint distribution of the two sentence spaces with one common hidden space, we also do not need any paraphrase data to train the network. Finally, our architecture is somewhat similar to (Lample et al. (2018); Rashid et al. (2019)) but they use a GAN-based model to distinguish a generated hidden vector from the source or target language instead of using a VAE framework.

2.2 Variational Auto-Encoders (VAEs)

We present in this Section the basics of Variational Auto-Encoder (VAEs) (Kingma & Welling (2014)).

2.2.1 Underlying Probabilistic Model

In the field of Machine Learning, we are usually interested in learning the probabilistic models underlying the data sets. Such mathematical representations, if we are able to find, are extremely useful for AI as they can perform a variety of tasks such as prediction and classification.

Suppose that we are given a dataset $\mathcal{D}$ consisting of n observed datapoints:

$$\mathcal{D} = \{\mathbf{x}_1, \ldots, \mathbf{x}_n\}.$$

The datapoints are assumed to be independent samples from an *unknown underlying distribution* $p(\mathbf{x})$, in other words, $\mathbf{x}_i \overset{i.i.d.}{\sim} p(\mathbf{x})$. The main goal is to approximate the true distribution $p(\mathbf{x})$ with a model $\mathbf{p}_\theta(\mathbf{x})$, with parameters θ :

$$\mathbf{p}_\theta(\mathbf{x}) \approx p(\mathbf{x}). \tag{2.1}$$

We suppose further that the variable $\mathbf{x}$, whose empirical distribution is usually complicated, is influenced by a *latent variable* $\mathbf{z}$, which we do not observe, in a hidden space $\mathcal{Z}$ whose distribution can be relatively simple.

To approximate this latent variable model, we use a *deep latent variable model* (DLVM):

$$\mathbf{p}_\theta(\mathbf{x}, \mathbf{z}) = \mathbf{p}_\theta(\mathbf{z})\mathbf{p}_\theta(\mathbf{x}|\mathbf{z}) \tag{2.2}$$

whose distributions are parameterized by neural networks, the $\mathbf{p}_\theta(\mathbf{z})$ and/or $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ are usually specified. The $\mathbf{p}_\theta(\mathbf{z})$ is often called the *prior distribution*.

However, the main difficulty of maximum likelihood learning in DLVMs is that in general the marginal probability of data under the model:

$$\mathbf{p}_\theta(\mathbf{x}) = \int \mathbf{p}_\theta(\mathbf{x}, \mathbf{z}) d\mathbf{z} \tag{2.3}$$

cannot be computed, or *intractable*, as it does not have an analytic solution or efficient estimator. Due to this intractability, we cannot simply differentiate the likelihood function

w.r.t. its parameters and optimize it. Note that since $\mathbf{p}_\theta(\mathbf{x})$ is intractable and $\mathbf{p}_\theta(\mathbf{x}, \mathbf{z})$ is tractable to compute, the identity:

$$\mathbf{p}_\theta(\mathbf{z}|\mathbf{x}) = \frac{\mathbf{p}_\theta(\mathbf{x}, \mathbf{z})}{\mathbf{p}_\theta(\mathbf{x})} \tag{2.4}$$

implies that the posterior distribution $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$ is also intractable to compute.

To solve this problem of intractability, we introduce a parametric *inference model* $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ with the aim of approximating the posterior distribution $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$:

$$\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x}) \approx \mathbf{p}_\theta(\mathbf{z}|\mathbf{x}). \tag{2.5}$$

This inference model is also parameterized using neural networks with parameters φ, also called the *variational parameters*.

A simple diagram with description of the whole VAE model is illustrated in Figure 2.1 (from Kingma & Welling (2019)). At this point of view, the model can be seen as an auto-encoder process where the inference model $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ acts as an encoder trying to map the distribution of $\mathbf{x}$-space to the distribution of $\mathbf{z}$-space and the generative model $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ acts as a decoder trying to remap the distribution of $\mathbf{z}$-space to the distribution of $\mathbf{x}$-space, hence the name Variational Auto-Encoder.

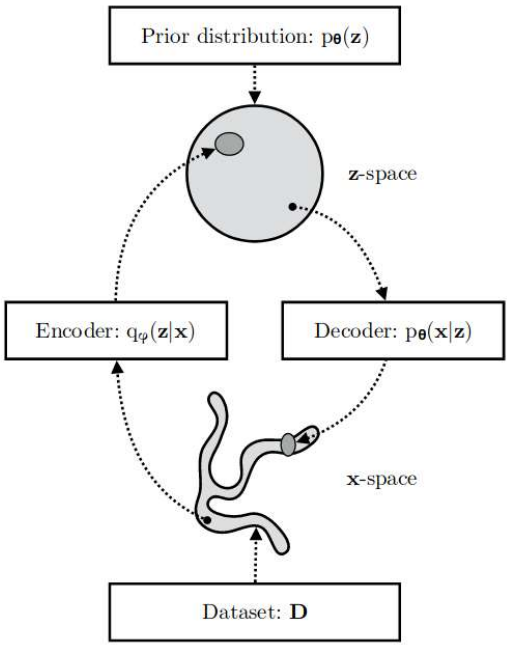

Figure 2.1: (from Kingma & Welling (2019)) A VAE learns stochastic mappings between an observed $\mathbf{x}$-space, whose empirical distribution $q_\mathcal{D}(\mathbf{x})$ is typically complicated, and a latent $\mathbf{z}$-space, whose distribution can be relatively simple (such as spherical, as in this figure). The generative model learns a joint distribution $\mathbf{p}_\theta(\mathbf{x}, \mathbf{z})$ often factorized as $\mathbf{p}_\theta(\mathbf{x}, \mathbf{z}) = \mathbf{p}_\theta(\mathbf{z})\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$, with a prior distribution over latent space $\mathbf{p}_\theta(\mathbf{z})$, and a stochastic decoder $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$. The stochastic encoder $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$, also called inference model, approximates the true but intractable posterior $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ of the generative model.

2.2.2 Loss Function of VAE

We will explain why the inference model $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ helps us train the generative model.

We first define a so-called *evidence lower bound* (ELBO), or *variational lower bound*:

$$\mathcal{L}_{\theta,\varphi}(\mathbf{x}) = \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \mathbf{p}_\theta(\mathbf{x},\mathbf{z}) - \log \mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})\right]. \tag{2.6}$$

We will show that the ELBO $\mathcal{L}_{\theta,\varphi}(\mathbf{x})$ is indeed a lower bound of the log marginal probability $\log \mathbf{p}_\theta(\mathbf{x})$, hence maximizing $\mathcal{L}_{\theta,\varphi}(\mathbf{x})$ also maximizes the marginal likelihood $\mathbf{p}_\theta(\mathbf{x})$.

Proposition 2.2.1. The log marginal probability $\log \mathbf{p}_\theta(\mathbf{x})$ is always lower-bounded by the ELBO $\mathcal{L}_{\theta,\varphi}(\mathbf{x})$, given any choice of inference model $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$, in other words:

$$\log \mathbf{p}_\theta(\mathbf{x}) \geq \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \mathbf{p}_\theta(\mathbf{x},\mathbf{z}) - \log \mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})\right]. \tag{2.7}$$

Proof. For any choice of inference model $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$, including the choice of variational parameter φ, we have:

$$\begin{aligned}
\log \mathbf{p}_\theta(\mathbf{x}) &= \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \mathbf{p}_\theta(\mathbf{x})\right] && (2.8)\\
&= \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{\mathbf{p}_\theta(\mathbf{x},\mathbf{z})}{\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})}\right]\right] && (2.9)\\
&= \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{\mathbf{p}_\theta(\mathbf{x},\mathbf{z})}{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\frac{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}{\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})}\right]\right] && (2.10)\\
&= \underbrace{\mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{\mathbf{p}_\theta(\mathbf{x},\mathbf{z})}{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\right]\right]}_{\mathcal{L}_{\theta,\varphi}(\mathbf{x})} + \underbrace{\mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}{\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})}\right]\right]}_{D_{KL}(\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})||\mathbf{p}_\theta(\mathbf{z}|\mathbf{x}))} && (2.11)\\
&= \mathcal{L}_{\theta,\varphi}(\mathbf{x}) + D_{KL}\left(\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})||\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})\right). && (2.12)
\end{aligned}$$

The second term in equation (2.12) is the Kullback-Leibler (KL) divergence between $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ and $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$. Since KL divergence is always non-negative, we deduce from equation (2.12) that:

$$\log \mathbf{p}_\theta(\mathbf{x}) \geq \mathcal{L}_{\theta,\varphi}(\mathbf{x}), \tag{2.13}$$

and the equation holds if and only if $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ equals the posterior distribution $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$. □

From equation (2.12), we also deduce that:

$$\mathcal{L}_{\theta,\varphi}(\mathbf{x}) = \log \mathbf{p}_\theta(\mathbf{x}) - D_{KL}\left(\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})||\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})\right). \tag{2.14}$$

Equation (2.14) infers that maximization of the function $\mathcal{L}_{\theta,\varphi}(\mathbf{x})$ w.r.t. the parameters θ and φ will concurrently maximize the marginal likelihood $\mathbf{p}_\theta(\mathbf{x})$ and minimize the KL divergence between the inference model $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ and the posterior distribution $\mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$, hence better approximate $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x}) \approx \mathbf{p}_\theta(\mathbf{z}|\mathbf{x})$.

2.2.3 A VAE Implementation

Now as we have prepared the theoretical ideas of VAE, we will present briefly how a typical VAE model is built.

First, in the encoder size, the inference model $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ is usually a normal distribution whose mean and variance are outputs of a neural network with weights φ and input $\mathbf{x}$:

$$\begin{aligned}(\boldsymbol{\mu}(\mathbf{x}), \log \boldsymbol{\Sigma}(\mathbf{x})) &= \text{NeuralNet } 1(\mathbf{x}; \varphi) \\ \mathbf{q}_\varphi(\mathbf{z}|\mathbf{x}) &= \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}), \text{diag}\left(\boldsymbol{\Sigma}(\mathbf{x})\right)\right).\end{aligned} \tag{2.15}$$

In the decoder size, the model $\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})$ is parameterized by an other neural network with weights θ:

$$\mathbf{p}_\theta(\mathbf{x}|\mathbf{z}) = \text{NeuralNet } 2(\mathbf{z}; \theta), \tag{2.16}$$

and the prior distribution $\mathbf{p}_\theta(\mathbf{z})$ is specified as the standard normal distribution:

$$\mathbf{p}_\theta(\mathbf{z}) = \mathcal{N}(\mathbf{z}; 0, \mathbf{I}). \tag{2.17}$$

Note that the loss function in equation (2.6) can be written as:

$$\mathcal{L}_{\theta,\varphi}(\mathbf{x}) = \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \mathbf{p}_\theta(\mathbf{x}, \mathbf{z}) - \log \mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})\right] \tag{2.18}$$

$$= \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \left[\mathbf{p}_\theta(\mathbf{x}|\mathbf{z})\mathbf{p}_\theta(\mathbf{z})\right] - \log \mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})\right] \tag{2.19}$$

$$= \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \mathbf{p}_\theta(\mathbf{x}|\mathbf{z})\right] - \underbrace{\mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\frac{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}{\mathbf{p}_\theta(\mathbf{z})}\right]} \tag{2.20}$$

$$= \mathbb{E}_{\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})}\left[\log \mathbf{p}_\theta(\mathbf{x}|\mathbf{z})\right] - D_{KL}\left(\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})||\mathbf{p}_\theta(\mathbf{z})\right). \tag{2.21}$$

The first term in equation (2.21) can be seen as a *negative reconstruction loss* in an auto-encoder model $\boldsymbol{\Delta}(\mathbf{x}, \widehat{\mathbf{x}})$ while the second one can be computed as, for $\mathbf{q}_\varphi(\mathbf{z}|\mathbf{x})$ and $\mathbf{p}_\theta(\mathbf{z})$ defined in equations (2.15) and (2.17):

$$D_{KL}\left(\mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}), \text{diag}\left(\boldsymbol{\Sigma}(\mathbf{x})\right)\right)||\mathcal{N}(\mathbf{z}; 0, \mathbf{I})\right) = \sum_{i=1}^{d} \frac{1}{2}\left(\mu_i^2 + \Sigma_i - \log(\Sigma_i) - 1\right), \tag{2.22}$$

where d is the dimension of the $\mathbf{z}$-space and $\boldsymbol{\mu}(\mathbf{x}) = (\mu_1, \ldots, \mu_d); \boldsymbol{\Sigma}(\mathbf{x}) = (\Sigma_1, \ldots, \Sigma_d)$.

To summarize, we can sketch the VAE model as follows:

$$\overbrace{\mathbf{x} \xrightarrow[\text{weights } \varphi]{NeuralNet1} (\boldsymbol{\mu}(\mathbf{x}), \log \boldsymbol{\Sigma}(\mathbf{x})) \rightarrow \mathbf{q}_\varphi(\mathbf{z}|\mathbf{x}) = \underbrace{\mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}), \text{diag}\left(\boldsymbol{\Sigma}(\mathbf{x})\right)\right)}_{\textbf{sampling}} \sim \mathbf{z}}^{\textbf{encoder}} \overbrace{\xrightarrow[\text{weights } \theta]{NeuralNet2} \mathbf{p}_\theta(\mathbf{x}|\mathbf{z}) \sim \widehat{\mathbf{x}},}^{\textbf{decoder}} \tag{2.23}$$

and the loss function is computed as:

$$\mathcal{L}_{\theta,\varphi}(\mathbf{x}) = \boldsymbol{\Delta}(\mathbf{x}, \widehat{\mathbf{x}}) - \sum_{i=1}^{d} \frac{1}{2}\left(\mu_i^2 + \Sigma_i - \log(\Sigma_i) - 1\right). \tag{2.24}$$

One last thing to note is that the sampling stage $\mathbf{z} \sim \mathcal{N}\left(\mathbf{z}; \boldsymbol{\mu}(\mathbf{x}), \text{diag}\left(\boldsymbol{\Sigma}(\mathbf{x})\right)\right)$ in equation (2.23) makes it difficult to differentiate and hence do the back-propagation process. As a consequent, a *reparameterization trick* is proposed, where $\mathbf{z}$ is written as:

$$\mathbf{z} = \boldsymbol{\mu}(\mathbf{x}) + \boldsymbol{\Sigma}(\mathbf{x}) \odot \mathbf{z}_\mathbf{I}, \tag{2.25}$$

where $\odot$ denotes the element-wise product and $\mathbf{z}_\mathbf{I} \sim \mathcal{N}(\mathbf{z}; 0, \mathbf{I})$. This makes the computational graph of the model direct and therefore we can differentiate through it to make back-propagation possible.

2.3 Dual S-VAE: Learning Distributed Representations of Sentence Spaces

We propose in this Section a method to generate new data by learning two stochastic mappings between the two sentence spaces and a latent space $\mathcal{Z}$ using two VAE frameworks.

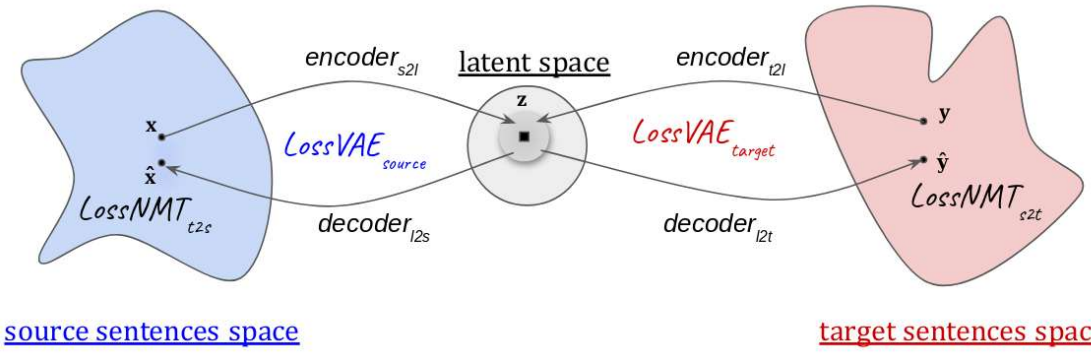

Figure 2.2: Illustration of the proposed model that combines a dual NMT model ($source \rightarrow target$ and $target \rightarrow source$) and two VAE frameworks (Dual S-VAE). The two encoders $s2l$ and $t2l$ map a sentence from source and target spaces to a distribution in the latent space, while the decoders $l2s$ and $l2t$ map each latent vector into source and target sentence spaces respectively. The $LossVAE_{source}$ and $LossVAE_{target}$ aim to learn the stochastic mappings between the two sentence spaces and the latent space, while the $LossNMT_{s2t}$ and $LossNMT_{t2s}$ try to align the source sentence space and the target sentence space.

Our method is illustrated in Figure 2.2. The model is composed of two encoders and two decoders. The $encoder_{s2l}$ and $encoder_{t2l}$ map a sentence from source and target spaces to a distribution in the latent space, while the $decoder_{l2s}$ and $decoder_{l2t}$ map each latent vector into source and target sentence spaces respectively. We denote:

$$decoder\left(encoder\left(s\right)\right) := decoder(h), h \sim encoder\left(s\right). \tag{2.26}$$

By that way $decoder_{l2s}\left(encoder_{s2l}\left(\mathbf{x}\right)\right)$ is the VAE-reconstructed sentence of $\mathbf{x}$ in the source space, while $decoder_{l2t}\left(encoder_{s2l}\left(\mathbf{x}\right)\right)$ is the stochastic translation of $\mathbf{x}$ in the target space.

The training of the model includes two main components:

Stochastic mappings via VAE. The model comprises two individual VAE frameworks. One maps the source sentence space to the latent space $\mathcal{Z}$ and remap the latent space to the source space. The other maps the target sentence space to $\mathcal{Z}$ and remap the latent space to the target space. The VAE loss is defined like in equation (2.21). More precisely, let $\mathbf{x}$ and $\mathbf{y}$ be sentences from source and target space respectively and $\mathbf{p}(\mathbf{z})$ is the prior distribution of the latent space, we have the two following VAE losses:

$$\mathcal{L}_{VAE_{source}} = \mathbf{\Delta}\left(\mathbf{x}, decoder_{l2s}\left(encoder_{s2l}\left(\mathbf{x}\right)\right)\right) - D_{KL}\left(encoder_{s2l}\left(\mathbf{x}\right)||\mathbf{p}(\mathbf{z})\right), \tag{2.27}$$

$$\mathcal{L}_{VAE_{target}} = \mathbf{\Delta}\left(\mathbf{y}, decoder_{l2t}\left(encoder_{t2l}\left(\mathbf{y}\right)\right)\right) - D_{KL}\left(encoder_{t2l}\left(\mathbf{y}\right)||\mathbf{p}(\mathbf{z})\right), \tag{2.28}$$

where $\mathbf{\Delta}(\cdot,\cdot)$ denotes the token wise cross-entropy loss between sentences.

Dual Learning NMT. Without the use of parallel data, the model only learns to map the two sentence spaces to the latent space but not between each other. In other words, given a latent vector $\mathbf{z}$, the two sentences $decoder_{l2s}(\mathbf{z})$ and $decoder_{l2t}(\mathbf{z})$ will not have the same meaning in the two languages. This encourages the use of parallel data to train the model to produce good translation of both ways ($source \rightarrow target$ and $target \rightarrow source$). The NMT losses are therefore defined, for two parallel sentences $\mathbf{x}$ and $\mathbf{y}$, as follows:

$$\mathcal{L}_{NMT_{s2t}} = \mathbf{\Delta}\left(\mathbf{x}, decoder_{l2t}\left(encoder_{s2l}\left(\mathbf{x}\right)\right)\right), \tag{2.29}$$

$$\mathcal{L}_{NMT_{s2t}} = \mathbf{\Delta}\left(\mathbf{y}, decoder_{l2s}\left(encoder_{t2l}\left(\mathbf{y}\right)\right)\right). \tag{2.30}$$

The final loss is defined as the weighted sum of the 4 losses :

$$\mathcal{L}_{DualS-VAE} = \lambda_{VAE}(\mathcal{L}_{VAE_{source}} + \mathcal{L}_{VAE_{target}}) + \lambda_{NMT}(\mathcal{L}_{NMT_{s2t}} + \mathcal{L}_{NMT_{s2t}}), \tag{2.31}$$

where λ_{VAE} and λ_{NMT} are hyper-parameters weighting the importance of the VAE and NMT losses respectively.

2.4 Experimental Results

2.4.1 Experimental Setup

Corpora. We choose a small corpus for our implementation. We take the publicly available English-French corpora[1]: *news commentaries* and *common crawl* containing 800K parallel pairs of sentences. We use the public dataset *News-test2008* as the validation data.

Network Configuration. All the encoders and decoders are built using self-attention architecture (Vaswani et al. (2017)) with 8 heads and 6 layers. We use 512 as the size of the word embeddings, and the size of each latent word vector is 100. We use the lazy Adam optimizer (Kingma & Ba (2015)). Learning rate is initially set to 0.001 and updated every 8 iterations. Before learning, we trained byte-pair encoding (BPE Sennrich et al. (2016c)) for each language using 30K merge operations each.

Word dropout and KL cost annealing. Bowman et al. (2016) points out a challenge for optimizing a VAE model for sequences, that is it usually sets the inference model $\mathbf{q}(\mathbf{z}|\mathbf{x})$ equal to the prior $\mathbf{p}(\mathbf{z})$, bringing the KL term in equation (2.21) to zero, which is not a correct behavior of a VAE model. A VAE model that encodes useful information in the latent variable $\mathbf{z}$ will have a non zero KL divergence term and a relatively small cross-entropy term. To overcome this issue, we apply two techniques proposed in the article. First, we perform a word dropout, which randomly delete each word in the input sentence with a given probability p before giving it to the encoder. Secondly, we put an additional weight in the KL divergence terms in equations (2.27) and (2.28) and set it to 0.001 in the first 10000 iterations so that the model can learn useful language information in the first steps.

[1] http://opus.nlpl.eu

2.4.2 Model Performance

We first asset the performance of our model by looking at the validation scores of the four directions: *source-to-source (s2s)*, *source-to-target (s2t)*, *target-to-target (t2t)*, *target-to-source (t2s)*. We also asset the effect of the hyper-parameters λ_{VAE} and λ_{NMT} to the performance of the model by varying different values of λ_{VAE}.

Model	s2s *(en2en)*	t2t *(fr2fr)*	s2t *(en2fr)*	t2s *(fr2en)*
Uni-directional NMT models	-	-	24.22	22.26
Dual S-VAE, $\lambda_{NMT} = 1$, $\lambda_{VAE} = 1$	58.19	51.65	9.42	9.28
Dual S-VAE, $\lambda_{NMT} = 1$, $\lambda_{VAE} = 0.5$	62.42	59.72	12.39	13.27
Dual S-VAE, $\lambda_{NMT} = 1$, $\lambda_{VAE} = 0.1$	68.70	66.22	17.98	18.33

Table 2.1: Performance of Dual S-VAE models (BLEU) in four directions: *source-to-source (s2s)*, *source-to-target (s2t)*, *target-to-target (t2t)*, *target-to-source (t2s)* using English→French dataset, compared to normal NMT systems.

Table 2.1 shows the BLEU score of our Dual S-VAE model compared with the scores of normal uni-directional NMT models English→French and French→English.

We can see that our model can reconstruct well source and target sentences, with BLEU scores (s2s, t2t) ranging from 50 to 70. This means that the model keeps most structures from the original sentences and in the same time introduces some modifications in reconstructed sentences as they are generated from random vectors. For the translation scores, we can see that our best model ($\lambda_{NMT} = 1$, $\lambda_{VAE} = 0.1$) gives BLEU scores nearly as high as the conventional NMT models in both directions even when the model performs a stochastic translation.

Comparing the different values of λ_{VAE}, we see that as we reduce the importance of VAE losses, all the NMT scores improve greatly.

2.4.3 Paraphrase Generation Task

We first examine the capacity of generating paraphrases of our model. We put a sentence from one language into the corresponding encoder to sample a latent vector, then this vector is put into the two decoders to reproduce either similar sentences to the first one or its translations.

Table 2.2 shows the examples when we put English sentences as input, while table 2.3 shows examples when using French input sentences. As we can see from the tables, the model can generate well the paraphrases of the original sentence, and even the paraphrases of its translation.

Reconstructed Sentences	Translated Sentences
Input: *The view from the hotel is beautiful.*	
The view of the hotel is beautiful.	La vue de l'hôtel est superbe.
The view from the hotel is nice.	La vue de l'hôtel est spectaculaire.
The view of the hotel is spectacular.	La vue sur l'hôtel est magnifique.
The atmosphere of the hotel is beautiful.	Le cadre de l'hôtel est magnifique.
The section of the hotel is lovely.	L'emplacement de l'hôtel est magnifique.
Input: *Local governments will manage the smaller enterprises.*	
Local governments will manage the smaller companies.	Les gouvernements locaux gèrent les petites entreprises.
Local governments will manage the smaller corporations.	Les gouvernements locaux doivent gérer les petites organisations.
Local governments will manage the lower businesses.	Les sociétés locales gèrent les petites entreprises.
Local governments will handle the smaller enterprises.	Les gouvernements locaux sont associés aux petites entreprises.
Local governments will support the smaller enterprises.	Les gouvernements locaux sont en mesure de gérer les petites entreprises.

Table 2.2: Examples of our model when producing paraphrases of English sentences and their stochastic translations in French. The outputs are taken from union of 5-best beam sets over different samplings, the English and French sentences are randomly placed (not aligned).

2.4.4 Parallel Generation Task

We test the capacity of generating parallel sentences of the model. We sample a random hidden vector in the latent code and put it to the two decoders to produce two sentences.

Table 2.4 demonstrates some examples of generated sentences. In this task, the model performs less well than the paraphrase generation task, with many pair of sentences that are not parallel. We suggest the reason is that if we sample a random hidden vector in the latent space, it will not usually produce a *proper vector* as if we sample it from the distribution after the encodage of a *real sentence*.

We regret, however, that we could not further asset the impacts of synthetic data produced by our model on the NMT as we focused more on the SCT approach that will be presented in the next chapter.

Reconstructed Sentences	**Translated Sentences**
Input: *Mon joli chien est très énergique.*	
Mon joli chien est très puissant.	My nice dog is very potent.
Mon charmant chien est très énergique.	My pretty dog is highly energetic.
Mon beau chien est très énergique.	My beautiful dog is highly potent.
Mon jolie chien est très vigoureuse.	My pretty dog is very vigorous.
Input: *Il est plus difficile de se cacher des ennemis.*	
Il est plus difficile de se masquer des ennemis.	It is more difficult to hide enemies.
Il est plus difficile de se dissimuler des ennemis.	It is harder to hide enemies.
Il est davantage difficile de se cacher des ennemis.	It's more difficult to hide enemies.
Il est également difficile de se cacher des ennemis.	It is more difficult to hide opponents.

Table 2.3: Examples of our model when producing paraphrases of French sentences and their stochastic translations in English. The outputs are taken from union of 5-best beam sets over different samplings, the English and French sentences are randomly placed (not aligned).

Generated English Sentences	**Generated French Sentences**
Acceptable Examples	
It is because of this respect.	C'est car le respect.
One older child or adult is charged 5.00 per night and person for extra beds.	Un enfant plus âgé ou adulte est facturé 20,00 EUR par nuit et par personne pour l'utilisation d'un lit d'appoint.
When would you like to stay at the hotel?	À quelle période souhaitez-vous séjourner dans cet établissement: camping?
Private parking.	Un parking privé.
The hotel is located in the island.	L'hôtel est installé à l'intérieur.
Bad Examples	
A 19th century old villa dating back to the 19th century.	Une rénovation de 1909 de Figueres a été pour vous.
As a result, you will be able to find an icon that will be displayed at the same time.	C'est depuis votre temps qu'on découvre l'illusion qu'il est vrai.
You can also contact us if you wish to travel with us.	Que vous soyez sur votre compte pour un séjour à votre goût, un conseiller ou un ami.
This is traditional and traditional education.	C'est le sud-est de l'Amérique du Sud.
Become around.	L'hôtel se trouve près de l'hôtel.

Table 2.4: Examples of our model when producing two parallel sentences from one hidden vector in the latent space.