Deep Variational Paraphrase Generation

Ta-Chung Chi (tachungc) [1] **Hira Dhamyal (hyd)** [1] **Prakhar Gupta (prakharg)** [1] **Yu-Hsuan Wang (yuhsuanw)** [1]

1. Introduction

Paraphrases are defined as sentences conveying the same meaning but with different surface realization. For e.g., for a sentence "How do I improve my English", a possible paraphrase is "What can I do to improve my English?". Both sentences have the same semantic meaning and intent. Paraphrases demonstrate the diverse nature of human language and paraphrase generation systems can be important components to other important language understanding tasks, such as question answering, machine translation or summarization by generating variants for automatic evaluation, and dialogue systems. Humans avoid using similar sentences when they have to repeatedly give the same information. Repetition makes an automated system less human-like as well as boring. Therefore a system capable of generating diverse paraphrases of any idea would be able to get better engagement.

Paraphrase generation can involve lexical and syntactic variation with differing degrees of semantic relatedness between the original sentence and the paraphrased one. Variations for paraphrasing can occur through various mechanisms and end-goals - a paraphrase may be optimized to yield precisely the same inferences as the original sentence, it can refer to the same entities in similar ways, with the same detail or can have varying level of lexical overlap between paraphrases. A lot of prior work in this area addressed these questions directly by exploiting linguistic knowledge, including handcrafted rules (Bolshakov & Gelbukh, 2004; Kauchak & Barzilay, 2006). These models had limited scope and were typically restricted in domain. They require considerable amount of human effort and do not generalize well, nonetheless produced paraphrases with good quality.

Many recent paraphrase generation systems focus on implicitly learning representations by training deep neural networks on datasets containing paraphrase pairs. Common approaches involve sequence-to-sequence (Sutskever et al., 2014) architectures prevalent in tasks such as machine translation. These approaches were followed by latent variable modelling based approaches which try to model various semantic and linguistics phenomena in text generation. Despite incremental improvements in these approaches and automatically scored metrics, the paraphrasing quality of state-of-the-art systems fall far short of what's needed for most applications. There are still challenges to be solved in these systems, such as making the systems more interpretable, controllable and able to achieve better diversity in the generation.

Variational auto-encoders (VAEs) are used in natural language generation as they regularize the latent space, and allow control and interpretability over the generation process. However, generating sentences from the continuous latent space does not explicitly model the syntactic information present in the sentences, as well as the semantic edits made in the sentence. In this project, we model the underlying factors for paraphrasing, by modelling semantics edits using either multiple latent variables or variational memory networks. We propose two approaches - 1) modelling the semantic edits in paraphrase as a latent actions. 2) disentangling syntactic and semantic spaces and leverage a memory module. We compare our models with various baselines and finally analyze the results of our models.

2. Literature Survey

Traditional methods for paraphrase generation generally use rule base content planning and surface realization procedures (Bolshakov & Gelbukh, 2004; Kauchak & Barzilay, 2006). These models have limited scope and fail to generalize to new instances. Recent approaches are primarily end-to-end models such as attentive seq2seq model (Su & Yan, 2017), the Residual LSTM model, the Gaussian VAE model (Gupta et al., 2018) and the transformer-based model (Li et al., 2019).

Deep Latent Variable Models - Deep latent variable models have found interesting applications in text modeling recently (Gupta et al., 2018; Fu et al., 2019). The latent variables model different properties of the text based on the task, such as alignment in translation, topics in dialogue systems (Serban et al., 2017), etc. One such work is (Gupta et al., 2018), where a standard VAE with a Gaussian prior is used. In contrast to the standard VAE, they additionally condition the encoder and decoder modules of the VAE on the original sentence. Fu et al. (2019) introduce a latent bag of words (BOW) model for paraphrase generation - the bag of words from the target sentences to ground the latent variable. They use source words to predict their neighbors

and model the target BOW with a mixture of softmax. They perform subset sampling from the predicted BOW distribution, retrieve the sampled word embeddings and use them in decoding to guide the decoder's generation search space.

Controllable Text Generation - Controllable text generation which allows users to control the generation process (Li et al., 2019; Hu et al., 2017). In Li et al. (2019), sentence paraphrasing is divided into two levels: sentence level and lexical level. Sentence level paraphrasing focuses on sentence structure transformation. The model consists of three modules: separator, two pairs of encoder-decoder and an aggregator. Given an input sentence, the separator would assign each word a tag specifying the word should be considered either sentence level or lexical level. Finally, two pairs of encoder-decoder generate two sequence of text and the aggregator merges them into the final generated sentence. The model is more interpretable since the corresponding transformation level for each word is traceable.

Edit based models (Huang et al., 2019) hypothesize that human write paraphrases by replacing words with their synonyms or phrases with close meanings . To model this behavior, they utilize an off-the-shelf dictionary to check for these substitutions and rewrite sentences into natural and consistent forms. The insertion and deletion operations can be explicitly modeled using the soft attention mechanism. This is inspired by Guu et al. (2018) that the model first proposes a prototype then rewrite it to the desired form.

Syntax guided paraphrasing - Recent works have shown that neural text generation can benefit from the incorporation of syntactic knowledge. Approaches include augmenting word representations with corresponding part-of-speech tag, lemmatized form and dependency label (Sennrich & Haddow, 2016), using a tree-based encoders and graph convolutional encoders to embed syntactic parse structures (Eriguchi et al., 2016), and leveraging implicit linguistic priors by treating syntactic parsing as an auxiliary task (Eriguchi et al., 2017). Zhang et al. (2019) proposed syntax-infused variation autoencoder architectures, leveraging constituency parse tree structure as the linguistic prior to generate more fluent and grammatical sentences. Their model generate more fluent and grammatical sentences. Chen et al. (2019b) leveraged linguistic and syntactical properties to improve paraphrase generation. Chen et al. (2019b) used additional losses in the training, designed to force the latent representations to capture different information. The semantic multi-task loss make use of aligned paraphrase data, whereas syntactic multi-task loss makes use of word-order information. This forces the latent space to disentangle and perform better.

3. Proposed Ideas and Methods

Recently, researchers have shown that explicitly syntactic modeling improves the generation quality in sequence-to-sequence models (Eriguchi et al., 2016; 2017). We explore adopting such ideas in the VAE setting. We first propose a edit prediction based multi-level VAE approach, followed by another approach to explicitly model the syntax and provide it as additional inductive bias to the model.

3.1. Approach 1: Edit Prediction

In this model, we leverage a multi-level VAE based model to predict sentence paraphrases. In normal VAE approach, the latent space encodes the semantics of the whole input sentence. Our approach is motivated by the fact that in paraphrase tasks, generally only some parts of the input are edited or modified. Our proposed model learns to predict the possible edit operations given the input sentence, and then conditions on the edit operation to generate the paraphrase. The architecture of the model is show in Figure 1 The model first predicts a latent variable z_x corresponding to which input sentence tokens the model will paraphrased. Then the model conditions on z_x to predict another latent variable z_y which the decoder uses to generate the final prediction.

The objective function of our model can be written as follows -

$$\begin{aligned}\mathcal{L}(\theta,\phi;\boldsymbol{x}_2,\boldsymbol{x}_1) =& \mathbb{E}_{q_\phi(\boldsymbol{z_y}|\boldsymbol{x}_1,\boldsymbol{x}_2)}\left[\log p_\theta\left(\boldsymbol{x}_2|\boldsymbol{z_y},\boldsymbol{x}_1\right)\right] \\ & -KL\left(q_\phi\left(\boldsymbol{z_y}|\boldsymbol{z_x},\boldsymbol{x}_1,\boldsymbol{x}_2\right)\|p(\boldsymbol{z_y}|\boldsymbol{z_x},\boldsymbol{x}_1)\right. \\ & -KL\left(q_\phi\left(\boldsymbol{z_x}|\boldsymbol{x}_1,\boldsymbol{x}_2\right)\|p(\boldsymbol{z_x}|\boldsymbol{x}_1)\right.\end{aligned}$$

Here x_1 is the original sentence and x_2 is the paraphrased sentence. We have two KL terms, one for z_x and another for z_y. The difference between these terms is that the z_y term is additionally conditioned on z_x. This objective function is based on Conditional Variational Encoder (Sohn et al., 2015) objective defined as -

$$\begin{aligned}\mathcal{L}(\theta,\phi;\boldsymbol{x}_2,\boldsymbol{x}_1) =& \mathbb{E}_{q_\phi(\boldsymbol{z}|\boldsymbol{x}_1,\boldsymbol{x}_2)}\left[\log p_\theta\left(\boldsymbol{x}_2|\boldsymbol{z},\boldsymbol{x}_1\right)\right] \\ & -KL\left(q_\phi\left(\boldsymbol{z}|\boldsymbol{x}_1,\boldsymbol{x}_2\right)\|p(\boldsymbol{z}|\boldsymbol{x}_1)\right)\end{aligned}$$

We additionally guide the first latent variable z_x to represent the positions of sentence which are paraphrased by an additional objective we call edit position loss $\mathrm{L_{edit}}$. To calculate this loss, z_x is concatenated with the LSTM encoder states of the sentence and a MLP layer is applied to that vector to get representations p_x for each encoder position. We then use a sigmoid activation for each position

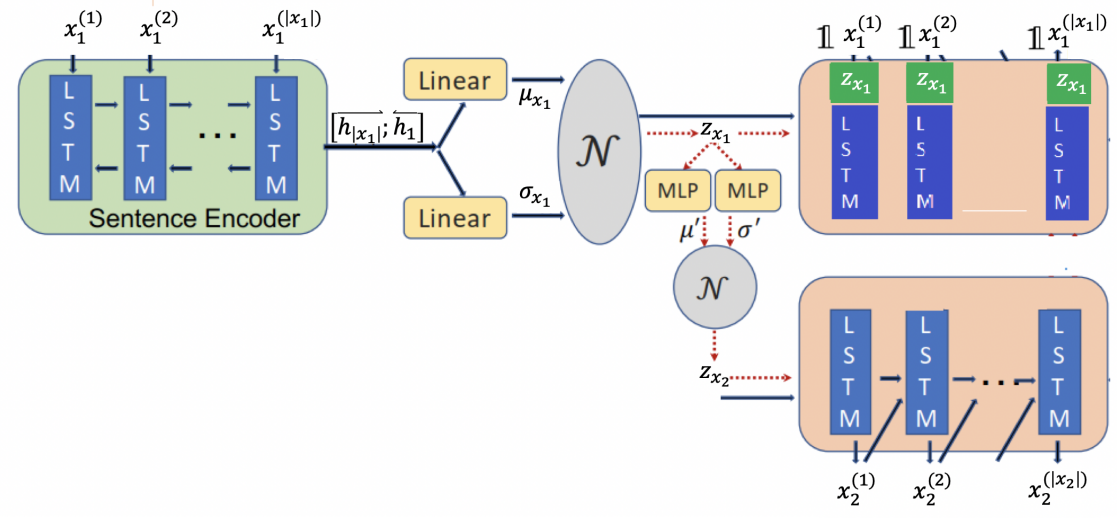

Figure 1. Approach 1: Edit Prediction: This model first predicts a latent variable z_x indicating which of the encoder positions are paraphrased. Then conditioned on this variable, z_y predicts the final representation which predicts the paraphrase

to predict the probability of editing the token at that position. We minimize the binary cross entropy loss between these probabilities and the actual labels of whether a token is paraphrased or not. z_y can be conditioned on z_x and the encoder states to finally predict the paraphrase sentence.

Edit loss term is defined as following -

$$\mathrm{L}_{\text{edit}} \stackrel{\text{def}}{=} \mathop{\mathbb{E}}_{z_x \sim q_\phi(z|x)} \left[-\sum_i \mathbb{L} \log \operatorname{softmax}\left(f\left([enc_i; z_x]\right)\right)_i \right] \tag{1}$$

We have two variations of our model - **Edit-Source** and **Edit-Target**. In Edit-Source, the latent variable z_x task is to predict the edit vector for the source sentence, and the enc_i vectors correspond to the encoded representations of the source sentence tokens. In Edit-Target, the latent variable z_x task is to predict the edit vector for the source sentence, and the enc_i vectors correspond to the encoded representations of the target sentence tokens. In Figure 2, we show how we prepare the label data for the Edit Loss. The cells with 0 indicate the positions which have been edited from source to target or vice-versa. The latent vector predicts the positions in the sentence which should be edited. During training of the model, we add the two losses defined above, that is the Elbo loss and the Edit loss.

Our two level latent variable approach thus adds additional structure and guidance to the original VAE objective by first learning what words can be paraphrased in a sentence, followed by the actual paraphrase generation.

3.2. Approach 2: Variational Memory Network

We propose to integrate a memory network component (Sukhbaatar et al., 2015) into a VAE architecture (Chen et al., 2019b), and modify the whole framework to be a CVAE-based model. Specifically, we use a memory network to explicitly model the syntax and semantics characteristic vectors of sentences.

3.2.1. CVAE-BASED DISENTANGLEMENT

Following (Chen et al., 2019a), our inference model disentangles a sentence $\mathbf{x}_1$ into two latent representations, syntax representation $\mathbf{z}$ and semantics representation $\mathbf{y}$. Our generation model uses $\mathbf{x}_1$ as well as the two representations to generate a corresponding paraphrase $\mathbf{x}_2$. The plate diagrams of the graphical model are depicted in Figure 3 and Figure 4.

We now derive the ELBO of approach 2. Recall that the empirical CVAE objective (Sohn et al., 2015) can be written as:

$$\begin{aligned}\mathcal{L}_{\text{CVAE}}(\mathbf{x_1}, \mathbf{x_2}; \theta, \phi) = &-KL(q_\phi(\mathbf{z_1}, \mathbf{y_1}|\mathbf{x_1}, \mathbf{x_2}) \| \\ &\quad p_\theta(\mathbf{z_1}, \mathbf{y_1}|\mathbf{x_1})) \\ &+ \frac{1}{L}\sum_{l=1}^{L} \log p_\theta\left(\mathbf{x_2}|\mathbf{x_1}, \mathbf{z_1}^{(l)}, \mathbf{y_1}^{(l)}\right)\end{aligned} \tag{2}$$

using our notations.

Our goal is to disentangle syntax and semantics, so it is

Source Sentence	what	problem	will	old	people	face	?		
Edit source	1	1	0	1	1	1	1		
Target Sentence	what	problems	do	old	people	have	to	face	?
Edit target	1	1	0	1	1	0	0	1	1

Figure 2. Sample Edit vector label preparation for Edit Loss calculation of Approach 1. The cells with 0 indicate the positions which have been edited from source to target or vice-versa.

reasonable to assume that $\mathbf{z_1}, \mathbf{y_1} \perp \mathbf{x_1}$, which leads to the factorization of the KL term as:

$$\int_{\mathbf{z_1}} \int_{\mathbf{y_1}} q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2) q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2) \log \frac{q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2) q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2)}{p_\theta(\mathbf{z}_1|\mathbf{x}_1) p_\theta(\mathbf{y}_1|\mathbf{x}_1)} d\mathbf{y}_1 d\mathbf{z}_1 \tag{3}$$

We expand the log function to get:

$$\int_{\mathbf{z_1}} \int_{\mathbf{y_1}} q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2) q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2) \log \frac{q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2)}{p_\theta(\mathbf{z}_1|\mathbf{x}_1)} + \int_{\mathbf{z_1}} \int_{\mathbf{y_1}} q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2) q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2) \log \frac{q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2)}{p_\theta(\mathbf{y}_1|\mathbf{x}_1)} \tag{4}$$

Note that $\mathbf{y_1}$ and $\mathbf{z_1}$ can be integrated out since they are valid probability distributions. Therefore, we have:

$$KL(q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2) \| p_\theta(\mathbf{z}_1|\mathbf{x}_1)) + KL(q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2) \| p_\theta(\mathbf{y}_1|\mathbf{x}_1)) \tag{5}$$

The latent variables $\mathbf{y}_1$ and $\mathbf{z}_1$ are assumed to be categorical. To be more specific, we set them as Bernoulli distributions.

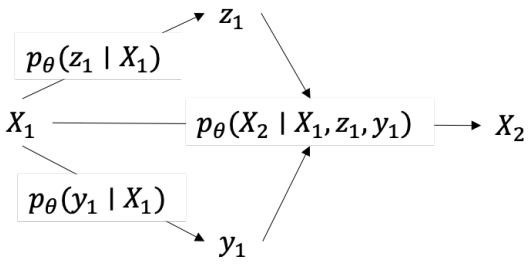

Figure 3. Generation model for the second approach. $\mathbf{x}_1$ and $\mathbf{x}_2$ are the input paraphrase. $\mathbf{z}_1$ and $\mathbf{y}_1$ are syntax and semantics latent codes for $\mathbf{x}_1$. Note that this can be viewed as two CVAEs where the target is $\mathbf{x}_2$

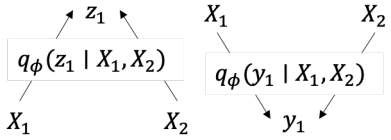

Figure 4. Inference model for the second approach. $\mathbf{x}_1$ is the input sentence. $\mathbf{z}_1$ and $\mathbf{y}_1$ are syntax and semantics latent codes, respectively.

3.2.2. Syntax and Semantic Representation

In this subsection, we describe how to model each components of the ELBO using neural networks. Given a sentence $\mathbf{x}_1$, its syntax representation $\mathbf{e}^{z_1}$ is encoded by *z-encoder* and the semantics representation $\mathbf{e}^{y_1}$ is encoded by *y-encoder*. Following (Chen et al., 2019a), we use a bi-directional LSTM as *z-encoder* and a feed forward neural network as *y-encoder*. The words for the input of *z-encoder* would be randomly replaced with another word having same POS tag since $\mathbf{e}^{z_1}$ should not be aware of any semantic information.

3.2.3. Generation with Memory Network

$\mathbf{e}^{z_1}$ and $\mathbf{e}^{y_1}$ are used to select candidate memory slots by computing dot product similarity with memory keys. The memory reading process is illustrated in Figure 6. The upper half of the figure, which is the syntax part, is used for demonstration. Concretely, the syntax latent code $\mathbf{z}_1 = [z_1^1, z_1^2, ..., z_1^K]$ is computed as

$$z_1^i = \sigma(\mathbf{k}_i^z \cdot \mathbf{e}^{z_1}), \tag{6}$$

where $\sigma(\bullet)$ is the sigmoid function and k_i^z is the key vector in the memory network.

We hypothesize the memory network encodes syntax transformation, where a key-value pair represents to transform from syntax A (key) to syntax B (value). Therefore, $\mathbf{z}_1$ can be viewed as K independent Bernoulli selections of new syntax which are used to reconstruct the sentence. In other words, we perform K Bernoulli sampling and use the sampled binary results $\bar{z}^1$ to select $\mathbf{v}_i$, which are the value

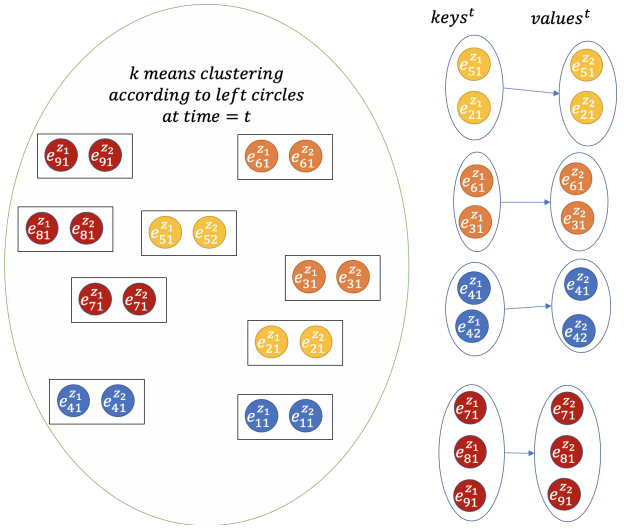

Figure 5. Memory network clustering.

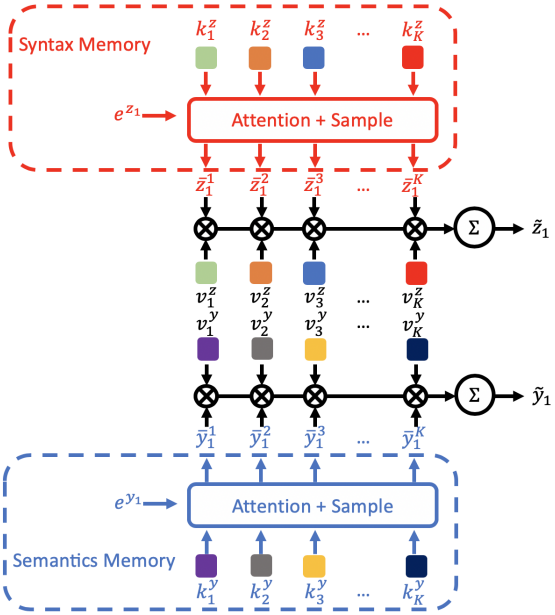

Figure 6. Memory network reading operation. $\mathbf{e}^{z_1}, \mathbf{e}^{y_1}$ are syntax representation and semantic representation for the input sentence $\mathbf{x}_1$, respectively. k_i and v_i are key-value pairs in the memory network.

vectors in our memory network:

$$\tilde{\mathbf{z}}_1 = \sum_{i=1}^{K} \tilde{z}_1^i \mathbf{v}_i^z \tag{7}$$

The semantics latent code $\mathbf{y}_1$ and $\tilde{\mathbf{y}}_1$ can be obtained in the same way. Finally, $\mathbf{e}^{z_1}$, $\mathbf{e}^{y_1}$, $\tilde{\mathbf{z}_1}$, $\tilde{\mathbf{y}_1}$ are concatenated for decoder to generate the paraphrase sentence $\mathbf{x}_2$. Since the decoder is provided with not only $\tilde{\mathbf{z}_1}$ and $\tilde{\mathbf{y}_1}$ but also $\mathbf{e}^{z_1}$ and $\mathbf{e}^{y_1}$, the estimated conditional distribution would be $p_\theta(\mathbf{x}_2|\mathbf{z}_1, \mathbf{y}_1, \mathbf{x}_1)$

The intuition behind of using both $\mathbf{e}^{z_1}$ and $\tilde{\mathbf{z}_1}$ is that $\tilde{\mathbf{z}_1}$ represents some specific syntax transformation operations and $\mathbf{e}^{z_1}$ represents the syntax information in $\mathbf{x}_1$. Based on the two information, the decoder would know perform what kinds of syntax transformation on $\mathbf{x}_1$.

3.2.4. Inference Model

The CVAE inference model generates the latent code with $\mathbf{x}_1$ and $\mathbf{x}_2$. With syntax and semantic encoders, we could get $\mathbf{e}^{z_1}$, $\mathbf{e}^{y_1}$, $\mathbf{e}^{z_2}$, $\mathbf{e}^{y_2}$ as illustrated in the previous section. We concatenate $\mathbf{e}^{z_1}$ and $\mathbf{e}^{z_2}$ and pass it through a feed forward neural network called *syntax-inference model* to get $\mathbf{e}^{z_1,z_2}$ and use it as the input of memory network to get the corresponding syntax latent code $\mathbf{z}_{1,2}$ and $\tilde{\mathbf{z}}_{1,2}$. On the other hand, we concatenate $\mathbf{e}^{y_1}$ and $\mathbf{e}^{y_2}$ and pass it through another feed forward neural network called *semantics-inference model* to get $\mathbf{e}^{y_1,y_2}$ and get $\mathbf{y}_{1,2}$ and $\tilde{\mathbf{y}}_{1,2}$ in the same way. $\mathbf{z}_{1,2}$ and $\mathbf{y}_{1,2}$ are treated as the inference distribution: $q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2)$ and $q_\phi(\mathbf{y}_1|\mathbf{x}_1, \mathbf{x}_2)$

3.2.5. Model Prior

In VAE, during the testing time the model has to sample latent code from a prior like Gaussian distribution for continuous variables or Bernoulli distribution for discrete variables. In CVAE, the prior could be any distribution based on the given condition. In our framework, the condition is $\mathbf{x}_1$. We use a feed forward network called *prior network* to transform the syntax representation $\mathbf{e}^{z_1}$ and semantics representation $\mathbf{e}^{y_1}$ into $\hat{\mathbf{e}}^{z_1}$ and $\hat{\mathbf{e}}^{y_1}$, respectively. $\hat{\mathbf{e}}^{z_1}$ and $\hat{\mathbf{e}}^{y_1}$ are then used to get latent codes and generate outputs. The latent codes $\mathbf{z}_1$ and $\mathbf{y}_1$ are treated as the conditional prior distributions: $p_\theta(\mathbf{z}_1|\mathbf{x}_1)$ and $p_\theta(\mathbf{y}_1|\mathbf{x}_1)$.

3.2.6. Memory Network Augmentation

The key value pairs in the memory network we used in previous sections comes from clustering of training data. The whole procedure is shown in Figure 5. Given a paraphrase training dataset of size N $\{\mathbf{x}_1^i, \mathbf{x}_2^i\}_{i=1}^N$ where $\mathbf{x}_1^i$ and $\mathbf{x}_2^i$ are the i-th paraphrase pair, we pass all sentences through the syntax/semantics encoder. For the syntax part this would be $\{\mathbf{e}_i^{z_1}, \mathbf{e}_i^{z_2}\}_{i=1}^N$ and we perform k means clustering using $\{\mathbf{e}_i^{z_1}\}_{i=1}^N$. The $\mathbf{k}_k^z$ is just the k-th cluster centers while we average the clustered $\{\mathbf{e}_i^{z_2}\}_{i=1}^N$ to get value vectors $\{\mathbf{v}_k^z\}_{k=1}^K$. Note that this step is computationally expensive so we plan to only perform re-clustering every fixed number of steps.

3.2.7. TRAINING

Given a sentence $\mathbf{x}_1$ and its paraphrase $\mathbf{x}_2$, we could get the corresponding syntax latent codes $\mathbf{z}_1, \mathbf{z}_2$, and semantics latent codes $\mathbf{y}_1, \mathbf{y}_2$.

The reconstruction term in the ELBO, or we call it the paraphrase loss, is:

$$\mathcal{L}_{para} = -\log p_\theta(\mathbf{x}_2|\mathbf{z}_1, \mathbf{y}_1, \mathbf{x}_1) \tag{8}$$

On the other hand, we need to minimize the KL-Divergence of latent variables and priors. Concretely, we use mean field approximation to approximate the inference network $q_\phi(\mathbf{z}_1|\mathbf{x_1}, \mathbf{x}_2)$:

$$\log q_\phi(\mathbf{z}_1|\mathbf{x_1}, \mathbf{x}_2) = \sum_{i=1}^{K} \log q_\phi(z_1^i|\mathbf{x_1}, \mathbf{x}_2) \tag{9}$$

The KL divergence term for syntax $\mathbf{z}_1$ could be computed as:

$$KL(q_\phi(\mathbf{z}_1|\mathbf{x}_1, \mathbf{x}_2)\|p_\theta(\mathbf{z}_1|\mathbf{x}_1)) = \sum_{i}^{K} q_\phi(z_1^i|\mathbf{x}_1, \mathbf{x}_2) \log\left[\frac{q_\phi(z_1^i|\mathbf{x}_1, \mathbf{x}_2)}{p_\theta(z_1^i|\mathbf{x}_1)}\right] \tag{10}$$

The KL divergence term for semantics $\mathbf{y}_1$ is computed in the same way.

In addition to the ELBO terms, we add another term for regularization. Recall that in section 3.2.3, $\tilde{\mathbf{z}}_1$ is designed to be the syntax transformation of $\mathbf{x}_1$. Therefore, if $\tilde{\mathbf{z}}_1$ is a zero vector, it would be mean that there is no syntax transformation. As a result, if we set $\tilde{\mathbf{z}}_1$ and $\tilde{\mathbf{y}}_1$ to be zero vectors, $\mathbf{0}_z$ and $\mathbf{0}_y$, and used them in the generation, the decoder should give us the original sentence $\mathbf{x}_1$. We call it the reconstruction loss $\mathcal{L}_{recon}$ as below:

$$\mathcal{L}_{recon} = -\log p_\theta(\mathbf{x}_1|\mathbf{0}_z, \mathbf{0}_y, \mathbf{x}_1) \tag{11}$$

The overall loss $\mathcal{L}$ for training is shown as below:

$$\begin{aligned}\mathcal{L} = \mathcal{L}_{para} + \mathcal{L}_{recon} \\ + \sum_{i}^{K} KL\left(q_\phi(z_1^i|\mathbf{x_1}, \mathbf{x}_2)||p_\theta(z_1^i|\mathbf{x_1})\right) \\ + \sum_{i}^{K} KL\left(q_\phi(y_1^i|\mathbf{x_1}, \mathbf{x}_2)||p_\theta(y_1^i|\mathbf{x_1})\right)\end{aligned} \tag{12}$$

where the KL divergence can be computed analytically and the reparameterization trick can also be derived (Jang et al., 2016; Maddison et al., 2016).

4. Experiments

In addition to implementing the baseline models and comparing them in the midterm report, here we will compare the baseline 1 and variants of it with our approach. (1) Baseline 1 is Deep Generative Framework for Paraphrased Generation(Gupta et al., 2018); which is a combination of LSTM and conditional VAE (where the generated sentence is conditioned on the original source sentence). (2) We implement a Conditional VAE (CVAE) model where in addition to the generation of the paraphrase being conditioned on the source sentence, the latent vector z is also conditioned on the source sentence. (3) We implement two versions of the Edit-prediction model as explained in section 3.1, where in (i) Edit-Source the task is for the edit vector to learn the edits in the source sentence and (ii) where the edit vector learns the edits in the target sentence. We notice that the edit-target achieves slightly better metrics in comparison to the edit-source, a reason for which could be that the paraphrase generated are more diverse. (4) We implement the Variational memory network as described in section 3.2.

4.1. Dataset

We perform experiments on the Quora duplicate question pairs which is one of the widely used corpus for paraphrasing. Quora dataset is annotated by humans and its size is around 130K pairs of sentences where we have 134,337 pairs for training and 14,928 pairs for testing.

4.2. Results and Analysis

Model	BLEU	Rouge-1	Rouge-2	Rouge-L
Baseline1	15.13	43.19	22.53	47.10
CVAE	14.62	45.73	24.12	49.11
Edit-Source	18.94	51.19	28.36	54.45
Edit-Target	19.05	51.36	28.68	54.61
Mem-Aug	20.47	50.01	28.20	53.47
Var-Mem	20.05	49.37	27.95	52.85

Table 1. Performance of Models on BLEU and Rouge

Baselines	METEOR	Syntax TED	TER
Baseline1	19.76	13.39	0.76
CVAE	23.51	15.23	0.87
Edit-Source	24.63	12.26	0.70
Edit-Target	24.67	12.08	0.70
Mem-Aug	27.25	11.31	0.68
Var-Mem	28.35	12.05	0.77

Table 2. Performance of Models on METEOR, Syntax TED and TER

The results table shows that Edit-Prediction model, specifically the edit-target, performs the best among all candi-

date models in terms of all variants of Rouge score. As for the Var-Mem model, it only performs better on METEOR. Even though our proposed methods do not outperform the baselines in all metrics, we can see that they are on par with the baselines.

5. Discussion

In this section, we will describe our key findings of the proposed methods. We want to highlight while quantitative results such as BLEU, rouge, or meteor are important, they sometimes cannot reflect the real capability of models (Mao & Lee, 2019). Hence, qualitative results will also be an important focus in the discussion.

5.1. Approach 1

5.1.1. Qualitative analysis

We analyze some of the outputs of the Edit-Source and Edit-Target models to understand which kind of sentences the model is not able to perform better in. Table 3 shows an example output of good paraphrase by the model. The Edit-Target model is able to replace the word 'short time' with 'fast'. Table 4 shows examples of some bad outputs by the model. In the first example, the model does not produce correct English grammar. In the second example, the model replaces the proper nouns in the source sentences. The current approaches need more improvements to resolve such issues.

Reference: how do i lose weight in a short time?
Edit Target: how do i lose weight fast?
Edit Source: how can i lose weight

Table 3. Example of Good paraphrase output.

Reference: what is the best way to learn how to write non-fiction?
Edit Target: how do i learn to write a new script
Edit Source: how do i write a fiction and write
Reference: would muhammad ali beat bruce lee?
Edit Target: who is better bruce lee or michael ali
Edit Source: who is better between bruce lee and ¡unk¿

Table 4. Example of Bad paraphrase output.

5.1.2. Edit Latent variable analysis

We perform an analysis where we keep the edit vector z_x fixed and vary z_y. We show some sample results in Table 5 for our Edit-Target model. We can see in the examples that the model fixes some part of the target sentence and paraphrases or adds other words, since we have fixed the edit latent variable z_x. For example, in the third row of the table, model keeps the following part of the target fixed - "how can i speak fluent english with" and then edits the words "fluently and fast" to either "confidence" or "accuracy". Similarily, in the second example, the model keeps the phrase "how much sleep" fixed and varies other words. Thus we can see that the edit vector is able to capture the edit operations.

Original sentence- do running increase your height?
Paraphrase 1: height : how would a 14 year old increase his height ?
Paraphrase 2: height : how would i increase my height ?
Original sentence- how much sleep do we really need as an adult in a night?
Paraphrase 1: how much sleep can adults have in a day ?
Paraphrase 2: how much sleep do we need to have a good night ?
Original sentence- how can i speak english fluently and fast?
Paraphrase 1: how can i speak fluent english with confidence ?
Paraphrase 2: how can i speak fluent english with accuracy ?

Table 5. Sample paraphrases by fixing z_x and varying z_y in Approach 1.

5.2. Approach 2

We first look at the quantitative results, which shows the performance of approach 2 is close to our best baseline. Next, we want to analyze the quality of disentangled representations, which are syntax and semantics, of the model. This can be demonstrated in two ways:

k-means-clustering Since there is a k-means-clustering step over the whole dataset during training, we can easily observe the generated clusters to see if there are any interesting effects. Concretely, we hypothesize that sentences with similar semantics or syntax should be clustered together. We put our qualitative results in Table 8 and 9. We can clearly see that in Table 8, the sentences in key 201 all begin with *what are*, and *is* in key 486. Ss for Table 9, the topic of key 398 is clearly about girls while they are about rupee in key 97.

latent space perturbation Recall that we have two vectors, $\mathbf{y}$ and $\mathbf{z}$, to represent semantics and syntax transformation. Therefore, we try to see the effect of them by turning one of each into a zero vector. For example, in Table 6, we can see that once the syntax vector is zeroed out, the generated output shares the same opening words as input. In Table 7, when the semantic vector is zeroed out, the generated sentence lacks information as in the original output.

input	i lost my password with my gmail account. how do i reset it?
output	how do i reset my gmail account without password or recovery no?
w\o syntax	i lost my gmail account password without a recovery information?

Table 6. Effect of removing syntax information.

input	how the way to speak english fluently?
output	how can i speak english properly and confidently?
w\o semantic	how can i speak english fluently?

Table 7. Effect of removing semantic information.

	syntax clusters
key 201	•what are the most beautiful and inexpensive countries to visit? •what are the best gadgets to buy on amazon? •what are good programming languages to learn in 2016?
key 486	•is it bad to be a sensitive person? •is donald trump in league with putin? •is drinking coca cola bad for health?

Table 8. k-means clustering result of syntax.

	semantic clusters
key 398	•how do i know she is into me? •how can i know girl is interested in me? •how do i tell if a girl i sit next to likes me?
key 97	•what are the repercussions of 500 and 1000 rupee notes not being legal? •what will be the effects of demonitizing 500 and 1000 rupees notes? •will rupee value rise after banning 500 and 1000 notes?

Table 9. k-means clustering result of semantic.

	baseline 3	approach 1	approach 2
#1st	86	78	86

Table 10. This is the result of human evaluation. We can see that all the approaches are pretty close with approach 1 being slightly worse.

5.3. Human Evaluation

We collect the first generated one hundred test sentences for human evaluation. Three models are compared in the test, which are baseline 3, approach 1, and approach 2. We recruit four students to pick the best paraphrase at their own discretion. For each question, we give the annotator an input sentence that is to be paraphrased. Then we randomly shuffle the generated sentences of the three models to reduce bias. The annotator is asked to pick the best paraphrase among three candidates. If they are all good or bad in a sense that is hard to distinguish, the annotator can put a zero and we skip the evaluation of that particular instance. The result is in Table 10. Our approaches are only comparable to the baseline, while approach 1 being a little bit worse.

6. Conclusion and Future Works

The proposed models could be explored further for better results. For example, in the current model we are not using the NER tags and we observe that in some of the predicted paraphrase sentences, the proper nouns in the sentence are getting changes like in the given example 4. We want the model to not change such words. Our idea is to not allow such changed in the edit vector for approach 1 and give the model that information during the key value mapping in the memory part of the model for approach 2. Our research is a step further in generating good paraphrase sentences, however, there is still room for improvement. As for the Var-Mem model, we think that the disentanglement can be further reinforced by introducing a adversarial loss (Bao et al., 2019). Specifically, given either syntax or semantics information, the model is not able to reconstruct the sentence. In addition, explicitly modeling the syntax structure of sentences by using parse trees may be another way (Bao et al., 2019).

Overall, we approach the paraphrase generation task using two unique approaches. Approach 1 tackles the problem by learning an 'edit vector' which learns which parts of the input sentences should be changed and which should be unchanged. In approach 2 we disentangle the syntax and semantics using a VAE model infused with a memory base network which utilizes a key value pair to model the syntax and the latent vector to capture the semantics. We show comparable results to the baseline models.

References

Bao, Y., Zhou, H., Huang, S., Li, L., Mou, L., Vechtomova, O., Dai, X., and Chen, J. Generating sentences from disentangled syntactic and semantic spaces. *arXiv preprint arXiv:1907.05789*, 2019.

Bolshakov, I. A. and Gelbukh, A. Synonymous paraphrasing using wordnet and internet. In *International Conference on Application of Natural Language to Information Systems*, pp. 312–323. Springer, 2004.

Chen, M., Tang, Q., Wiseman, S., and Gimpel, K. Controllable paraphrase generation with a syntactic exemplar. *arXiv preprint arXiv:1906.00565*, 2019a.

Chen, M., Tang, Q., Wiseman, S., and Gimpel, K. A multi-task approach for disentangling syntax and semantics in sentence representations. *arXiv preprint arXiv:1904.01173*, 2019b.

Eriguchi, A., Hashimoto, K., and Tsuruoka, Y. Tree-to-sequence attentional neural machine translation. *arXiv preprint arXiv:1603.06075*, 2016.

Eriguchi, A., Tsuruoka, Y., and Cho, K. Learning to parse and translate improves neural machine translation. *arXiv preprint arXiv:1702.03525*, 2017.

Fu, Y., Feng, Y., and Cunningham, J. P. Paraphrase generation with latent bag of words. In *Advances in Neural Information Processing Systems*, pp. 13623–13634, 2019.

Gupta, A., Agarwal, A., Singh, P., and Rai, P. A deep generative framework for paraphrase generation. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 2018.

Guu, K., Hashimoto, T. B., Oren, Y., and Liang, P. Generating sentences by editing prototypes. *Transactions of the Association for Computational Linguistics*, 6:437–450, 2018.

Hu, Z., Yang, Z., Liang, X., Salakhutdinov, R., and Xing, E. P. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1587–1596. JMLR. org, 2017.

Huang, S., Wu, Y., Wei, F., and Luan, Z. Dictionary-guided editing networks for paraphrase generation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 6546–6553, 2019.

Jang, E., Gu, S., and Poole, B. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*, 2016.

Kauchak, D. and Barzilay, R. Paraphrasing for automatic evaluation. In *Proceedings of the main conference on Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics*, pp. 455–462. Association for Computational Linguistics, 2006.

Li, Z., Jiang, X., Shang, L., and Liu, Q. Decomposable neural paraphrase generation. *arXiv preprint arXiv:1906.09741*, 2019.

Maddison, C. J., Mnih, A., and Teh, Y. W. The concrete distribution: A continuous relaxation of discrete random variables. *arXiv preprint arXiv:1611.00712*, 2016.

Mao, H. and Lee, H.-y. Polly want a cracker: Analyzing performance of parroting on paraphrase generation datasets. *arXiv preprint arXiv:1908.07831*, 2019.

Sennrich, R. and Haddow, B. Linguistic input features improve neural machine translation. *arXiv preprint arXiv:1606.02892*, 2016.

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

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Su, Y. and Yan, X. Cross-domain semantic parsing via paraphrasing. *arXiv preprint arXiv:1704.05974*, 2017.

Sukhbaatar, S., Weston, J., Fergus, R., et al. End-to-end memory networks. In *Advances in neural information processing systems*, pp. 2440–2448, 2015.

Sutskever, I., Vinyals, O., and Le, Q. V. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pp. 3104–3112, 2014.

Zhang, X., Yang, Y., Yuan, S., Shen, D., and Carin, L. Syntax-infused variational autoencoder for text generation. *arXiv preprint arXiv:1906.02181*, 2019.