Leveraging Pretrained Transformers for Paraphrase Generation

(Sixten Heekin, Oisin Turbitt, Anton Adar)

Abstract

Paraphrase generation is an established task in natural language processing. The applications include pre-processing for different tasks, injecting natural variation in conversation systems or data augmentation. Symbolic approaches are limited in the variation of the phrases they can produce. We present a conditional variational autoencoder (CVAE) framework to generate paraphrases by utilising pretrained transformer models (GPT-2 and RoBERTa). The results indicate that using these models provides an advantage over CVAEs which rely only on training long short-term memory (LSTM) networks. Furthermore GPT-2 produces slightly better results than RoBERTa in this framework. However, the paraphrases generate still suffer from grammatical inaccuracies and the semantic variations are hard to control. Further improvements are suggested in order to further develop the model.

1. Introduction

Paraphrase generation is a well established task in natural language processing. The goal of the task is that when given an input sentence, an output sentence is automatically produced that captures the same meaning. For instance, a paraphrase of the question "Why do birds have beaks?" might be "How come birds have beaks?". A robust paraphrase generation engine could serve many useful functions. For one, it could be a feature in a flexible and natural-feeling chat-bot application- particularly in the case where sentences may be paraphrased in a simpler and more appropriate style. Paraphrasing can also be used to summarize (Woodsend & Lapata, 2012) and compress longer pieces of text into a shorter version or with different style (Pavlick & Nenkova, 2015). It is also a crucial component of question answering systems (Fader et al., 2013). Furthermore, paraphrases can serve a more technical application. Data augmentation, the synthesis of new training data by applying slight variations to existing instances, is a widespread practice in training computer vision models employing some form of noise, rotation or crop to an image. Similarly, using paraphrases to augment data in natural language settings may thereby improve models generalization (Kumar et al., 2019).

Despite its potential usefulness, effective paraphrase generation has proven quite challenging to convincingly execute. In contrast to continuous data, there is no trivial way to induce minor variation to language, as it is discrete, non-numerical and highly context-dependent. Symbolic approaches to this problem involved replacing words with synonyms (Jing, 1998) or building rule sets to change sequences (Pang et al., 2003) (for example change "it is" to "it's") using finite state automata. However, synonym replacement alone does not produce a rich variation in the paraphrase. Constructing rule sets to produce such variation has proven to be a difficult proposition, as many rules are highly context dependent, meaning that after aggregating an initial basic and wide-ranging set, adding more rules only brings minor benefits, as further rules apply to specific situations. Therefore, symbolic solutions seem to be limited by either the flexibility of generation or the complexity involved with storing rules for rich paraphrase variation.

Another method involves searching through various translations of the same text in order to find sentences where translators differ in their translation, thereby obtaining different versions of the same sentence. This issue here is that these models are not particularly good at paraphrasing sentences that have not been translated before, and thus are constrained in their ability to handle cases that have not been added to the database (Ganitkevitch et al., 2013).

Some efforts at generating paraphrases have utilized existing neural machine translation methods in order to perform back-translation (Prabhumoye et al., 2018). This method involves translating a given sequence in one language into another language, and then translating back into the original language. The idea of this method is that the inacurracies and ambivalence in translation introduces variance in the original sentence. The measure of deviation of the output from the input sentence may be controlled by using beam search over translations with different likelihood. As a rule a less likely back-translation will produce a paraphrase which resembles the original sentence less closely. However, a less likely translation will often be a lower quality translation, meaning it may simply violate syntactic and semantic constraints. Therefore, there is a need to explicitly model the distribution of sentences and sample a paraphrase from that latent space on the basis of the original sentence.

Recently, several deep learning methods have been proposed specifically for paraphrase generation. These methods can be broken down into three different modelling categories: formulating the task as sequence-to-sequence problem;, employing reinforcement learning techniques; or employing generative model architectures. Methods based off sequence-to-sequence (seq2seq) cast paraphrase

generation task as a translation task from and to the same language. Using the experience of neural machine translation (Bahdanau et al., 2014), proposed architectures have employed stacked residual long short-term memory networks (LSTMs)(Prakash et al., 2016), transformer networks (Egonmwan & Chali, 2019), and gated recurrent units (GRUs) (Cao et al., 2017) in order to complete the paraphrase task. Models that rely on reinforcement learning have two networks, a generator and an evaluator (Li et al., 2017). The generator, based off a seq2seq model, attempts to create a paraphrase given an input sentence while the evaluator, consisting of a deep neural network, attempts to learn to tell if the output and input sentences are paraphrases.

This paper focuses on using generative model architectures and in particular using conditional variational autoencoders (CVAE) (Sohn et al., 2015). It follows on from previous work where two stacked LSTM's are used as both the encoder and decoder (Gupta et al., 2018). The output hidden state of the encoder is used to produce the latent variable which is then passed to the decoder. Each input to the decoder is then conditioned on the sampled output of the latent space in order to generate a paraphrase from a given input. However, this method produces results which are far from ideal as the outputs frequently are not coherent sentences.

We attempt to improve upon this by utilising pretrained transformer language models as an encoder. The sentence embedding produced by this encoder is then used to create the latent space from which we sample. Following (Gupta et al., 2018), this is then used to condition the inputs to the decoder in order to generate a paraphrase.

2. Background

2.1. Variational Autoencoders

A Variational Autoencoder (VAE) approximates probabilistic functions $p_\theta(z)$ and $q_\phi(x|z)$ parameterized by θ and ϕ in order to estimate a probability distribution over x. These functions are nonlinear transformations which take a given input x and produce (encode) a distribution over a compressed latent representation $p(z)$ such that when a sample z from this distribution is decoded, a sequence x^* similar to the original is reproduced. VAEs excel at taking high-dimensional data and generating rich and low dimensional representations (Kingma & Welling, 2013). Crucially, rather than autoencoders, VAEs don't learn deterministic representations of a given x, but rather a posterior distribution.

$$\begin{aligned} log(p_\theta(x)) = &-D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) \\ &+ D_{KL}(q_\phi(z|x) \parallel p_\theta(z|x)) \\ &+ \mathbb{E}_{q_\phi(z|x)}[log p_\theta(x|z)] \end{aligned} \quad (1)$$

As KL divergence is non-negative we can estimate a lower bound on $log p_\theta(x)$

$$log(p_\theta(x)) \geq -D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) + \mathbb{E}_{q_\phi(z|x)}[log p_\theta(x|z) \quad (2)$$

This lower bound on $log p_\theta(x)$ can serve as a lower bound of the objective function, Leading to the formulation of the corresponding loss function, sampling L points from latent space.

$$\mathfrak{L}(x;\theta,\phi) = -D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L} log p_\theta(x|z_l) \quad (3)$$

As the true prior $p_\theta(z)$ is not know, so it is assumed as a standard normal distribution $z \sim N(0,1)$. This furthermore allows us to reparameterize the variational encoder to deterministically produce the parameters of a normal distribution, from which samples may be drawn by injecting noise, reframing the encoder as a function $g_\phi(x,\epsilon)$ with a deterministic component x and a random component ϵ. This means that the VAE can be properly trained using gradient descent. Finally, similar to a VAE, the CVAE will generate from the latent space may be decoded conditionally by providing a conditioning input y to the decoder, thereby changing the generative process to leverage both z and y, $q_\theta(x|z,y)$. Therefore the new reconstruction objective and loss is given by 2.1.

$$\mathfrak{L}_c(x,y;\theta,\phi) = -D_{KL}(q_\phi(z|x) \parallel p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L} log p_\theta(x|z_l,y) \quad (4)$$

2.2. Leveraging Pre-trained Transformers

Many advances in recent years in natural language processing and understanding have leveraged transformer architectures (Vaswani et al., 2017), which, in contrast to recurrent architectures, uses attention as its main processing mechanism. In an transformer attention layer there is a mechanism to 'attend' over a fixed number n of input vectors, by producing a weighed sum over linear projections of the input vector. The so called query and key vectors determine weights.

$$\vec{z} = \sum_{i=1}^{n} softmax(\vec{q}_i \vec{k}_i^T)\vec{v}_i \quad (5)$$

These three vectors are produced by linear projections of the inputs given by weights W_q, W_k, W_v. Intuitively, the query and key vectors weigh the input, while the projection to the value vector learns features from the input vector much like a fully connected layer. Additionally, transformers typically feature several different sets of weights for each of these representations, creating several independent attention processes in the same layer, which are referred to as heads. The final output is the concatenation of each of the heads' outputs. Therefore a transformer layer is able to process an input representation with a flexible weight on each input vector, determined by different objectives. As

this dispenses with most of the recurrent architecture it has allowed more extensive parallelization of the calculation. Compared to recurrent architectures, which only process one input element at each time point, it has the flexibilty to attend over all inputs.

The advent of transformers was accompanied by the publication of powerful language models pre-trained on a variety of tasks. These models are able to be adapted for a multitude of different natural language processing and understanding tasks, with only very little training and adaptation of the architecture. However, we are not aware of any experimentation of pre-trained transformers as VAEs.

The Bidirectional Encoder Representations from Transformer (BERT) (Devlin et al., 2018) has a deeply bi-directional architecture, meaning it processes a sequence right-to-left as well as left-to-right simultaneously. It was trained as a masked language model, filling in obfuscated words in a sequence, as well as next-sentence prediction. The Robustly optimized BERT approach (RoBERTa) (Liu et al., 2019) refines this pre-training procedure and is trained on a larger dataset. By adding a small number of layers, or training the last layers of BERT on other tasks the original pre-trained model was adapted to perform natural language understanding tasks, such as logic inference, sentiment classification, as well as Question Answering outperforming state-of-the-art models by a considerable margin.

The second iteration of the Generative Pre-Training model (GPT-2) (Radford et al., 2019) also follows a transformer architecture, although it is set up to be a unidirectional (left-to-right) architecture. It was exclusively trained as a language model. The model was initially adapted to perform natural language understanding and question answering tasks.

There are two reasons that the pre-trained transformer models mentioned above might be suitable for the paraphrasing task explored in this paper. The first reason hinges on the the attention mechanism in the transformer architecture. As mentioned above, language is both discrete and highly interdependent. This means that replacing a single a word in a sentence might have a drastic effect on the semantics of that sentence, since the coherence of many other words might depend on that particular word. Thus an effective paraphrase generator must be able to capture all the dependencies between different words in a sentence, in order to produce a string that has the same semantic content as the original text. As discussed above, the attention mechanism utilized in transformer architectures such as BERT and GPT-2 is designed to capture exactly these kinds of dependencies. Thus it is conceivable that pre-trained transformers might be a useful feature of a paraphrase generating neural network, since they could encode information about which in a given piece of text could be adjusted without affecting too much change in the overall meaning of the text.

The next advantage of pre-trained transformer architectures is the vast quantity of data on which they are trained and the huge number of parameters which they learn. BERT, for example, was trained on a dataset including most of the internet as well as hundreds of millions of words from a plethora books. This huge quantity of data, coupled with model that can learn an extremely complex function, allows BERT and GPT-2 to capture much of the nuance and diversity present in language. This is useful for paraphrase generation because good paraphrases can range from subtly adjusted versions of the original text all the way to versions that are almost entirely different in terms of syntax and diction. Thus making use of a model that has been trained on an incredibly broad range of language usage might be useful when attempting paraphrase generation, in the same way that is has proved useful for performing other NLP tasks.

Thus we expect both RoBERTa and GPT-2 too produce rich embeddings suitable for the paraphrasing task. However, they do not follow a probabilistic model, which is why we they need to be adjusted to fulfill this criterion. In order to answer our research question regarding whether pretrained transformers make good variational encoders for paraphrase generation, we propose these hypotheses:

1. *Both adapting either RoBERTa or GPT-2 as variational encoders will lead to better paraphrase quality in a CVAE framework than using an LSTM variational encoder.*

We reason that the GPT-2 pretraining task of language modelling is more closely aligned with the task of generating paraphrases. Therefore we predict:

2. *GPT-2 will produce better paraphrases in a CVAE framework than RoBERTa.*

3. Model

For a given sentence s^O, and an example paraphrase s^P, a dataset S can be made consisting of N examples, $(s_n^O, s_n^P)_{k=1}^N$. The aim of this model is to use S in order learn a function which will, when given s_n^O, perform a mapping that will generate s_n^P. The model, shown in Figure 1, consists of three units: a variational encoder, an exact encoder and a paraphrase decoder.

The sentences first have to be converted into a vector from via an embedding mechanism. Every sentence consists of a words such w_t is the t-th word in a sentence of length T. A word embedding, $\mathbf{W_e} \in \mathbb{R}^{d \times V}$ where V is the vocabulary size, is used to convert each word of a sentence into a d-dimensional vector representation, $\mathbf{x_t}$ such that $\mathbf{x_t} = \mathbf{W_e}[w_t]$. A matrix representation of a sentence can then be made from all word vectors of that sentence, $\mathbf{X} \in \mathbb{R}^{d \times T}$. The model will use these representations in order to generate $\mathbf{X}^P$ from a latent space $\mathbf{z}$ and $\mathbf{X}^O$.

Encoders Both encoders take $\mathbf{X}^O$ as input. The variational encoder $q_\phi(\mathbf{z}|\mathbf{x}^O)$ consists of a pretrained transformer model

which produces a sentence embedding representation. This is then fed through a feedforward network in order to generate μ and σ of a Gaussian distribution $\mathcal{N}(\mu, \sigma)$. Using the reparameterization trick, we can then sample from $\mathbf{z}$. This is then passed as to decoder. A LSTM is used for the exact encoder which takes $\mathbf{X}^O$ and passes its final hidden and cell states onto the decoder.

Decoder The decoder $p_\theta(\mathbf{x}^P|\mathbf{z}, \mathbf{x}^O)$ is an LSTM network which attempts to predict a paraphrase sentence s'^P by generating each word $w_t'^P$ in sequentially until the end of sentence token is generated. At each t-th step, the sampled latent variable $\mathbf{z}$ is concatenated at every step to the input of the decoder. The input is the previous ground truth word w_{t-1}^P in a technique known as input feeding (Luong et al., 2015). The word vector representation for this is generated by an additional word embedding $\mathbf{W_d} \in \mathbb{R}^{d \times V}$. Each predicted word is generated by greedily sampling from a probability distribution given by $w_t'^P = argmax(\mathbf{V h_t})$ where $\mathbf{V}$ is trained linear layer and $\mathbf{h_t}$ is the hidden state of the decoder at t. The final states of the deterministic encoder for the initialisation of the decoder states at step 0, while $\mathbf{x}_0^P$ is the word vector of the start of sentence token.

Implementation The model is trained end-to-end in a supervised fashion via back propagation. This is achieved by maximising the loss function given by:

$$\mathfrak{L}_c(x, y; \theta, \phi) = -D_{KL}(q_\phi(z|x^O) \parallel p_\theta(z)) + \frac{1}{L}\sum_{l=1}^{L} log\, p_\theta(x^P|z, x^O) \tag{6}$$

Training a VAE for text is a difficult task and so several tricks where used to improve training. (Bowman et al., 2016) recommends using two techniques. The first is called KL annealing which involves adding a weight to the KL loss term which initially starts at 0 and increases to 1 over a set number of training iterations. The second technique randomly replaces the previous ground truth word with the token representing an unknown word. Following further recommendations in (Bowman et al., 2016) and in (Yang et al., 2019), highway networks (Srivastava et al., 2015) are used between the word embedding layers and the LSTM models and between the variational encoder and the decoder. These help with the backpropagation of gradients through the model. One particular difficultly of the model implementation arises from the fact that the lengths of s^O and s^P do not have be equal. Because transformers produce an embedding output for each word in a sentence it is not possible take this output and pass it through the network. Instead, a universal sentence embedding approach (Reimers & Gurevych) is used by taking the mean of sentence embedding produced by a transformer and passing this matrix through the network in order to generate $\mathbf{z}$.

4. Experiments

4.1. Model Setup

In order to create and train the models, the Fairseq [1] (Ott et al., 2019) and HuggingFace [2] (Wolf et al., 2019) libraries. These are dedicated platform for natural language processing models built on top of PyTorch (Paszke et al., 2019) which allow for use of pretrained models. Two variations of the model were tested using different variational encoders consisting the base version of pretrained RoBERTa and GPT-2 transformer models. RoBERTa base is a transformer type architecture with 12 layers with 12-head attention and a hidden dimension of 768, amounting to 125M parameters. GPT-2 base has 12 layers with a hidden dimension of 768 and 12 heads amounting to 117M parameters. During training time all the weights of the pretrained transformers were frozen in order to keep both training time and the number of parameters manageable.

For the LSTM encoder, a bidirectional 2 layer network was used with a hidden size of 512. The LSTM decoder consisted of a unidirectional 2 layer network with with an attention mechanism using the inputs of the LSTM encoder (Luong et al., 2015). A latent space size of 1024 was used. The 300 dimensional Glove pretrained embeddings were used to initialise the embeddings of the LSTMs (JeffreyPennington & Manning, 2014). All highway networks used 2 layers with a Gaussian error linear unit activation function (Hendrycks & Gimpel, 2016). A word dropout rate of 0.25 was used for the decoder and dropout layers set at a probability of 0.3 used throughout the network (Srivastava et al., 2014). The weight of the KL loss was increased linearly from 0 to 1 over the first 750 training iterations.

The models were trained using a batch size of 1024 for 50 epochs or until an early stopping criteria was achieved, which was if there was no decease in validation loss for 3 epochs. Nearly all experiments ran hit this stopping criteria first and during test time, the model used was the model from training with the lowest validation loss. The large batch size was achieved using an initial batch size of 256 and using the update frequency option prodived by Fairseq to simulate running on 4 GPUS. The AdamW optimiser (Loshchilov & Hutter, 2017) was used, with an initial learning rate set at 0.001 which was linearly decreased to 0.0001 over of duration of training. A weight decay condition of 0.01 was also set along with a gradient clipping norm of 2 (Pascanu et al., 2013)

Sentences were tokenized using byte-pair encodings (Sennrich et al., 2015). This technique splits words into subwords units, which allows for effectively smaller dictionaries while attenuating issues regarding rare words. Specifically we employed the tokenizer used by RoBerta and GPT2 (Radford et al., 2019), with a vocabulary size of 50k and produced vector representation a 786 dimensional embedding.

[1] https://github.com/pytorch/fairseq
[2] https://github.com/huggingface/transformers

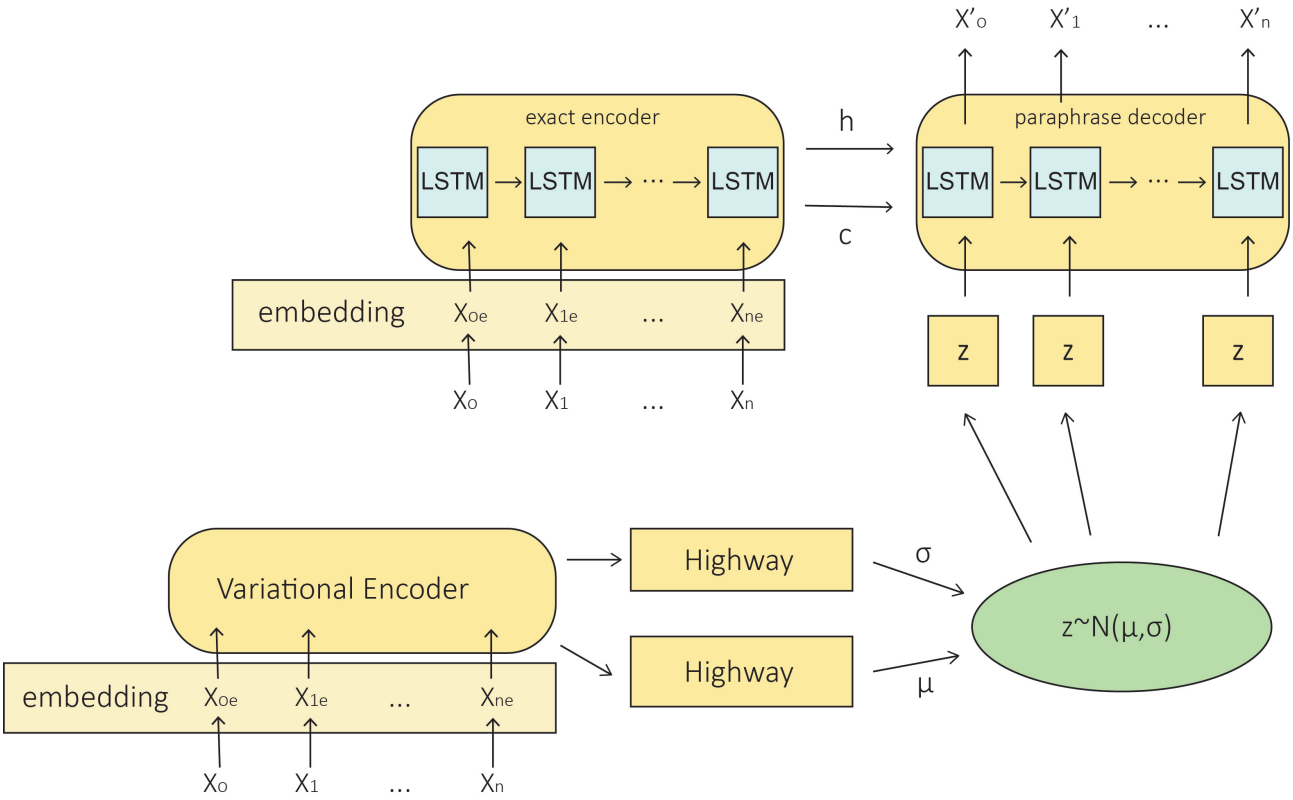

Figure 1. Our model architecture. The variational encoder is implemented using a pretrained transformer model. In our experiements, we focused upon RoBERTa or GPT-2

4.2. Dataset and Evaluation

The Quora dataset of 800k question pairs was used as our dataset [3]. This dataset was originally published as a kaggle challenge to identify duplicate questions. It has since become an established dataset for paraphrase generation as it contains labelled examples of paraphrases. Of the 150K question pairs marked as being identical questions we selected 104k question pairs to use training set and 22k for validation and test set each.

For quantitative evaluation we calculate the BLEU (Papineni et al., 2002) scores of all paraphrases. The bilingual evaluation understudy score (BLEU) is a metric from neural machine translation rated out of 100 where 100 represents the exact same sentence. It scores a pair of sentences in similarity by counting how many token sequences of a given length n (n-grams) occur in both sentences. In the example "Why do birds have beaks?" and "How come birds have beaks?", the tri-grams "birds have beaks" occurs in both sentences, contributing to a high BLEU score, but "do birds have" only occurs in the first sentence lowering the BLEU score.

4.3. Baselines

A VAE was created and trained in a supervised fashion to act as a baseline. Here use the model network described above but with the pretrained transformer encoder removed. Instead the decoder is conditioned using the final hidden cells of the LSTM encoder. A second baseline is also reported for the VAE-SVG-eq model using the results from (Gupta et al., 2018) for the model trained 100K Quora paraphrases.

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

5. Results

The results for the BLEU score on the test along with the number of model parameters are shown in table 1. Example paraphrases are also provided in table 2. There was a clear gap in performance between the baselines compared our model with the pretrained transformers. However, the framework using GPT-2 performed worse compared to the RoBERTa model, albeit with only a small margin. Further improvements in BLEU scores could also be achieved by additional tuning of hyperparameters.

6. Discussion

The one core advantage of CVAEs in this task is being able to generate several different paraphrases based on the same condition of the original sentence. However, we found only little variation in the paraphrases the model produced. We reason that this issue could be addressed by more carefully modelling the variational encoder, to ensure it represents the natural variation of language more closely. One measure could be to weigh the divergence term of the loss more strongly. We wanted to leverage RoBERTa and GPT2 as

MODEL	TEST BLEU	PARAMETERS (TRAINED)
VAE-S	23.62	59M (59M)
VAE-SVG-EQ	22.9	70M (70M)
ROBERTA	27.95	203M (79M)
GPT-2	26.74	195M (79M)

Table 1. Performance of the frameworks with different variational encoders, along with the number of parameters and trained parameters

Original Sentence	How is black money curbed with the ban of 1000 rupee notes and introducing new 500 and 2000 rupee notes?
VAE-S Paraphrases	How will the ban on 500 and 1000 rupee notes stop black money?
RoBERTa Model Paraphrases	How will the ban on 500 and 1000 rupee note stop black money? How will the India demonetization of 500 and 1000 rupees notes will reduce black money?
GPT-2 Model Paraphrases	How will the India demonetization of 500 and 1000 rupees notes will reduce black money? How will the ban on 500 and 1000 rupee note stop black money?

Original Sentence	How is black money curbed with the ban of 1000 rupee notes and introducing new 500 and 2000 rupee notes?
VAE-S Paraphrases	How will the ban on 500 and 1000 rupee notes stop black money?
RoBERTa Model Paraphrases	How will the ban on 500 and 1000 rupee note stop black money? How will the India demonetization of 500 and 1000 rupees notes will reduce black money?
GPT-2 Model Paraphrases	How will the India demonetization of 500 and 1000 rupees notes will reduce black money? How will the ban on 500 and 1000 rupee note stop black money?

Original Sentence	What's your feeling on the fact Hillary Clinton won popular vote while losing the election?
VAE-S Paraphrases	What do you think of the first presidential election?
RoBERTa Model Paraphrases	What are your views on the popular vote for Hillary Clinton despite losing the election? What are your views on the fact that Hillary Clinton won the election?
GPT-2 Model Paraphrases	Why did the Clinton vote for Hillary Clinton despite losing the election? Why did the Clinton vote for Hillary Clinton despite losing the election?

Table 2. Example generated paraphrases using test set examples of original sentences. These were produced by trained models of the described architecture with either a RoBERTa or GPT-2 as the variational encoder. A generated paraphrase from the VAE-S baseline model is also provided for comparison

transfer learning tools, which produced good results. The original idea is to use their rich representation in order to solve a specific natural language task using only very low training time and little data does not apply so easily to this frame work. While Devlin et al (Devlin et al., 2018) report fine-tuning a question answering system for only 30mins, incorporating these models into a CVAE, the adaptation is not as simple. This task requires bot a deterministic encoder and a decoder model to be trained. Parameterizing this model even very conservatively, the model complexity quickly increases. One large contributor is the final output projection from hidden states to token likelihood. As vocabulary size typically ranges from 10k-100k, this layer easily contributes parameters in the order of 10^7. Reducing the size of the hidden states more aggressively will definitively accelerate the training. However, this will naturally limit the models performance. So while a pre-trained model might only need minor adaptation for many tasks, in our case we still had to train a considerable number of 80 million parameters.

6.1. Further Work

It can be seen in the results that using pretrained transformers to condition a VAE can lead to significant improvements in BLEU score. However further work is required to validate these results. Additional evaluation metrics, such as METEOR (Lavie & Agarwal, 2007) or TER (Snover et al., 2006), should be used to further test the model. The model should then be trained and tested upon other available paraphrase datasets. In literature, these usually are the Microsoft Paraphrase Corpus (Dolan & Brockett, 2005) and the Microsoft Common Objects in Context dataset image labels (Lin et al., 2014). Further improvements could also be obtained by using a beam search (Wiseman & Rush, 2016) instead of greedy word selection when picking an output word w'^P.

Improvements to the architecture and training scheme were also identified in preliminary tests but needed to be further developed. One of these was to condition the variational encoder on s^O and s^P such that $q_\phi(\mathbf{z}|\mathbf{x}^O)$, which would align the model architecture closer to (Gupta et al., 2018) and (Yang et al., 2019). It would also be of interest to keep the weights of the pretrained transformer unfrozen in order to fine tune the network for paraphrase generation. This

proved fruitful in our experiements, however more testing was required. Investigating the change in evaluation scores of using the larger model variations for the pretrained transformers would also be of interest.Another idea would be to remove all LSTM and utilise transformers for deterministic encoder and decoder like what is done in (Egonmwan & Chali, 2019).

7. Related Work

There have been a few other papers dealing explicitly with deep generative approaches to paraphrase generation. The first is "Generating Sentences from a Continuous Space" by Samuel Bowman et al. This paper explores the latent space of a variational autoencoder composed of two LSTMs as decoder and encoder. In doing so, they found that by sampling from the latent space, coherent sentences could be generated. However, these authors did not experiment explicitly with paraphrase generation, nor did they leverage transformer architectures as they had not been created yet (Bowman et al., 2016).

The next paper that deals in a similar area to our own is "Deep Generative Framework for Paraphrase Generation" by Ankush Gupta et al. This paper utilizes a similar architecture as the previous one, leveraging both RNNs and VAEs. However they note that traditional VAEs are not suitable for paraphrase generation, even though they can be used to generate text. In order to remedy this issue, they condition both the encoder and decoder on the original text in order to generate more reliable paraphrases. Our experiment incorporates this technique as well, albeit with the inclusion of pre-trained transformers in our architecture (Gupta et al., 2018).

Another paper that deals with generative approaches to paraphrase generation is "An End-to-End Generative Architecture for Paraphrase Generation" by Qian Yang et al. This paper explicitly takes its architectural ques from the previously discussed paper: it utilizes a VAE conditioned on the inputs. However its departure point is the inclusion of a GAN and a discriminator that forces the model to learn representations that more closely align to the structure of the source sentences (Yang et al., 2019).

Lastly, "Transformer and seq2seq for Paraphrase Generation" by Elozino Egonmwan et al. In this paper the authors utilize a traditional autoencoder with a transformer architecture implemented as part of the encoder section. This transformer is used to create a sentence embedding that is passed on as a hidden state of an LSTM that serves as another encoder layer. This LSTM produces a hidden-state that is then decoded into a paraphrase deeper in the network. However these researchers trained their own transformer instead of using a pretrained model, and did not utilize a VAE. (Egonmwan & Chali, 2019).

8. Conclusion

This project aimed to generate paraphrases to a given sentence, reflecting the same meaning in a different linguistic form. We chose the framework of conditional generation as it matches two characteristics of the problem, it introduces natural variance, while being based on the condition of the original sentence. Leveraging pre-trained transformers for this task has shown to improve performance considerably, although both models we selected did not differ much. We consider these results to be an indication to the power and flexibility of transfer learning in a natural language processing context. That being said, we do not reckon that our experiment leveraged the full capacity of these tools. Implementing additional changes to architecture elements and hyper-parameters may improve performance even further.

References

Bahdanau, Dzmitry, Cho, Kyunghyun, and Bengio, Yoshua. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

Bowman, Samuel R., Vilnis, Luke, Vinyals, Oriol, Dai, Andrew, Jozefowicz, Rafal, and Bengio, Samy. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pp. 10–21, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/K16-1002. URL https://www.aclweb.org/anthology/K16-1002.

Cao, Ziqiang, Luo, Chuwei, Li, Wenjie, and Li, Sujian. Joint copying and restricted generation for paraphrase. In *Thirty-First AAAI Conference on Artificial Intelligence*, 2017.

Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. 10 2018. URL http://arxiv.org/abs/1810.04805.

Dolan, William B and Brockett, Chris. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*, 2005.

Egonmwan, Elozino and Chali, Yllias. Transformer and seq2seq model for paraphrase generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pp. 249–255, Hong Kong, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5627. URL https://www.aclweb.org/anthology/D19-5627.

Fader, Anthony, Zettlemoyer, Luke, and Etzioni, Oren. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1608–1618, 2013.

Ganitkevitch, Juri, Durme, Benjamin Van, and Callison-Burch, Chris. Association for Computational Linguistics PPDB: The Paraphrase Database. Technical report, 2013. URL http://paraphrase.org.

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

Hendrycks, Dan and Gimpel, Kevin. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*, 2016.

JeffreyPennington, RichardSocher and Manning, ChristopherD. Glove: Global vectors for word representation. Citeseer, 2014.

Jing, Hongyan. Usage of wordnet in natural language generation. In *Usage of WordNet in Natural Language Processing Systems*, 1998.

Kingma, Diederik P and Welling, Max. Auto-encoding variational bayes, 2013.

Kumar, Ashutosh, Bhattamishra, Satwik, Bhandari, Manik, and Talukdar, Partha. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pp. 3609–3619, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1363. URL https://www.aclweb.org/anthology/N19-1363.

Lavie, Alon and Agarwal, Abhaya. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pp. 228–231, Prague, Czech Republic, June 2007. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/W07-0734.

Li, Zichao, Jiang, Xin, Shang, Lifeng, and Li, Hang. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279*, 2017.

Lin, Tsung-Yi, Maire, Michael, Belongie, Serge, Bourdev, Lubomir, Girshick, Ross, Hays, James, Perona, Pietro, Ramanan, Deva, Zitnick, C. Lawrence, and Dollár, Piotr. Microsoft COCO: Common Objects in Context. may 2014. URL https://arxiv.org/abs/1405.0312.

Liu, Yinhan, Ott, Myle, Goyal, Naman, Du, Jingfei, Joshi, Mandar, Chen, Danqi, Levy, Omer, Lewis, Mike, Zettlemoyer, Luke, and Stoyanov, Veselin. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019. URL http://arxiv.org/abs/1907.11692.

Loshchilov, Ilya and Hutter, Frank. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Luong, Minh-Thang, Pham, Hieu, and Manning, Christopher D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Ott, Myle, Edunov, Sergey, Baevski, Alexei, Fan, Angela, Gross, Sam, Ng, Nathan, Grangier, David, and Auli, Michael. fairseq: A fast, extensible toolkit for sequence modeling. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pp. 48–53, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-4009. URL https://www.aclweb.org/anthology/N19-4009.

Pang, Bo, Knight, Kevin, and Marcu, Daniel. Syntax-based alignment of multiple translations: Extracting paraphrases and generating new sentences. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pp. 102–109, USA, 2003. Association for Computational Linguistics. doi: 10.3115/1073445.1073469. URL https://doi.org/10.3115/1073445.1073469.

Papineni, Kishore, Roukos, Salim, Ward, Todd, and jing Zhu, Wei. Bleu: a method for automatic evaluation of machine translation. pp. 311–318, 2002.

Pascanu, Razvan, Mikolov, Tomas, and Bengio, Yoshua. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pp. 1310–1318, 2013.

Paszke, Adam, Gross, Sam, Massa, Francisco, Lerer, Adam, Bradbury, James, Chanan, Gregory, Killeen, Trevor, Lin, Zeming, Gimelshein, Natalia, Antiga, Luca, et al. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems*, pp. 8024–8035, 2019.

Pavlick, Ellie and Nenkova, Ani. Inducing lexical style properties for paraphrase and genre differentiation. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 218–224, 2015.

Prabhumoye, Shrimai, Tsvetkov, Yulia, Salakhutdinov, Ruslan, and Black, Alan W. Style transfer through back-translation, 2018.

Prakash, Aaditya, Hasan, Sadid A, Lee, Kathy, Datla, Vivek, Qadir, Ashequl, Liu, Joey, and Farri, Oladimeji. Neural paraphrase generation with stacked residual lstm networks. *arXiv preprint arXiv:1610.03098*, 2016.

Radford, Alec, Wu, Jeff, Child, Rewon, Luan, David, Amodei, Dario, and Sutskever, Ilya. Language models are unsupervised multitask learners. 2019.

Reimers, Nils and Gurevych, Iryna. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. Technical report. URL https://github.com/UKPLab/.

Sennrich, Rico, Haddow, Barry, and Birch, Alexandra. Neural machine translation of rare words with subword units. *CoRR*, abs/1508.07909, 2015. URL http://arxiv.org/abs/1508.07909.

Snover, Matthew, Dorr, Bonnie, Schwartz, Richard, Micciulla, Linnea, and Makhoul, John. A study of translation edit rate with targeted human annotation. In *In Proceedings of Association for Machine Translation in the Americas*, pp. 223–231, 2006.

Sohn, Kihyuk, Lee, Honglak, and Yan, Xinchen. Learning structured output representation using deep conditional generative models. In Cortes, C., Lawrence, N. D., Lee, D. D., Sugiyama, M., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems 28*, pp. 3483–3491. Curran Associates, Inc., 2015. URL http://papers.nips.cc/paper/5775-learning-structured-output-representation-using-deep-conditional-generative-models.pdf.

Srivastava, Nitish, Hinton, Geoffrey, Krizhevsky, Alex, and Salakhutdinov, Ruslan. Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Technical report, 2014.

Srivastava, Rupesh Kumar, Greff, Klaus, and Schmidhuber, Jürgen. Highway networks. *arXiv preprint arXiv:1505.00387*, 2015.

Vaswani, Ashish, Shazeer, Noam, Parmar, Niki, Uszkoreit, Jakob, Jones, Llion, Gomez, Aidan N., Kaiser, Lukasz, and Polosukhin, Illia. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Wiseman, Sam and Rush, Alexander M. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960*, 2016.

Wolf, Thomas, Debut, L, Sanh, V, Chaumond, J, Delangue, C, Moi, A, Cistac, P, Rault, T, Louf, R, Funtowicz, M, et al. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv, abs/1910.03771*, 2019.

Woodsend, Kristian and Lapata, Mirella. Multiple aspect summarization using integer linear programming. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pp. 233–243, Jeju Island, Korea, July 2012. Association for Computational Linguistics. URL https://www.aclweb.org/anthology/D12-1022.

Yang, Qian, Huo, Zhouyuan, Shen, Dinghan, Cheng, Yong, Wang, Wenlin, Wang, Guoyin, and Carin, Lawrence. An end-to-end generative architecture 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)*, pp. 3132–3142, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1309. URL https://www.aclweb.org/anthology/D19-1309.