Paraphrase Generation via Adversarial Penalizations

Gerson Vizcarra and **José Ochoa-Luna**
Department of Computer Science
Universidad Católica San Pablo
Arequipa, Perú
{gerson.vizcarra,jeochoa}@ucsp.edu.pe

Abstract

Paraphrase generation is an important problem in Natural Language Processing (NLP) that has been addressed with neural network-based approaches recently. This paper presents an adversarial framework to address the paraphrase generation problem in English. Unlike previous methods, we employ the discriminator output as penalization instead of using policy gradients, and we propose a global discriminator to avoid the Monte-Carlo search. In addition, this work use and compare different settings of input representation. We compare our methods to some baselines in the Quora question pairs dataset. The results show that our framework is competitive against the previous benchmarks.

1 Introduction

Paraphrase generation is a task in NLP which aims to transform a given sentence in another with the same meaning. This task is challenging because of the complexity of semantic and syntactic relationships in language. Moreover, the capacity to generate paraphrases automatically is an opportunity to use data augmentation in NLP. However, one of the main problems of paraphrase generation is that the meaning of a sentence can be changed radically by modifying a word.

Paraphrase generation is a hot task and recent neural-approaches have addressed the problem. We organize previous works of paraphrase generation into two groups: task-support and task-based. The task-support paraphrase generation works create paraphrases to add training data or adversarial-examples. Paraphrases have been created to augment data in question answering (Dong et al., 2017; Gan and Ng, 2019). Some techniques are substitution of words (Jiao et al., 2019; Xie et al., 2019), make syntactic changes to the original sentences (Coulombe, 2018; Iyyer et al., 2018; Sennrich et al., 2016), and back-translation (Mallinson et al., 2017; Xie et al., 2019). Other works apply rule-based generative models to perform multiple changes (Samanta and Mehta, 2017; Li et al., 2017).

On the other hand, task-based paraphrase generation works aim to benchmark their results on specific paired datasets. In this paper, we focus on task-based works. Prakash et al. (2016) use a stacked residual Long-Short Term Memory (LSTM) network that outperforms a vanilla LSTM. Gupta et al. (2018) apply a Variational Autoencoder (VAE) to get better results than the stacked LSTM. Huang et al. (2018) use a Seq2Seq-based model with a dictionary-based attention mechanism. The dictionary search guides the insertion or deletion of a word. Li et al. (2018b) generate paraphrases using a deep reinforcement learning framework. Ma et al. (2018) propose an attention network using word embeddings information. Yang et al. (2019) present an adversarial setup over latent space using a conditional VAE as a generator. Chen et al. (2019) propose a VAE to make syntactically and semantically changed paraphrases. Wang et al. (2019) propose a transformer (Vaswani et al., 2017) with multiple encoders to process extra semantic information of the input. The work of Egonmwan and Chali (2019) shows a hybrid model between a transformer and a Recurrent Neural Network (RNN). Li et al. (2019) design a transformer-based model that can generate paraphrases at different levels of granularity.

From the prior works on paraphrase generation, most of them learn by conditional Maximum Likelihood Estimation (MLE). However, Yang et al. (2019) highlight the exposure bias problem (Bengio et al., 2015; Ranzato et al., 2015) in paraphrase generation. Some works address the problem by applying REINFORCE (Williams, 1992) to generate text in their adversarial setups (Yu et al., 2017; Fedus et al., 2018; Li et al., 2018a; Liu et al., 2018; de Masson d'Autume et al., 2019). However, similar to prior works (He et al., 2019; Lu et al., 2019), we observe that REINFORCE has a high variance and is difficult to tune.

At the same time, pre-trained Language Models (LMs) (Devlin et al., 2019; Peters et al., 2018; Radford et al., 2019) have outperformed previous works in many NLP tasks. An important reason is that they provide contextual representations of words, which are more specific than static word embeddings (Pennington et al., 2014; Mikolov et al., 2013, 2018). However, static word embeddings consume fewer resources and are faster than pre-trained LMs. Ethayarajh (2019) shows that the static embeddings extracted from pre-trained LMs outperform Glove (Pennington et al., 2014) and FastText (Bojanowski et al., 2017) in many word vector benchmarks.

In this paper, we propose an adversarial model (generator-discriminator) to address the English paraphrase generation task. Unlike previous approaches, we train our model using a weighted conditional maximum likelihood by a "penalization" score given by the discriminator. Also, we test variations of our setup by changing the input representations and the Monte-Carlo search. Overall, our contributions are as follows.

- We propose the use of penalizations (discriminator outputs) in supervised adversarial setups as an alternative to the REINFORCE algorithm.

- We evaluate the substitution of the Monte-Carlo search by using a discriminator that outputs a score for each word.

- We provide an experimental analysis of the impact of input representations over a paraphrase generation model. Further, we include the use of the first-layer embeddings from pre-trained language models.

- Our experiments show that our setup can generate feasible paraphrases. Furthermore, our results are competitive against prior benchmarks in the Quora question pairs dataset.

2 Preliminaries

Before presenting our model, we provide some preliminaries about the MLE training and REINFORCE algorithm in sequential problems.

2.1 Conditional Maximum Likelihood Estimation

The conditional MLE training for sequence to sequence tasks aims to learn the probability distribution of the estimated token $\hat{y}_t$ constrained by an input sequence $X_{1:T} = \{x_1, ..., x_T\}$ and a list of previous tokens $\hat{Y}_{1:t-1} = \{\hat{y}_1, ..., \hat{y}_{t-1}\}$. We consider the sets of tokens X and $\hat{Y}$ in a finite vocabulary V. Given a dataset D with pairs of sequences of tokens $X_{1:T}, Y_{1:T}$ with length T (due to truncation and padding), the objective function J_{MLE} of conditional MLE is

$$J_{MLE} = \mathbb{E}_{\hat{Y}\sim Y}[\sum_{t=1}^{T} \log G(\hat{y}_t|X, \hat{Y}_{1:t-1})] \quad (1)$$

Where $G(\hat{y}_t|\hat{Y}_{1:t-1}, X)$ is the neural network that generates the word $\hat{y}_t$. Most works use teacher forcing (Williams and Zipser, 1989) by using the correct tokens Y instead of $\hat{Y}$ to improve the results. In that way, the "exposure" of the network to the target words could "bias" the inference process.

2.2 REINFORCE

REINFORCE (Williams, 1992) in sequence to sequence tasks aims to maximize the reward r of a policy (neural network) received due to the generation of the word $\hat{y}_t$. In similar dataset conditions, the objective function J_R of REINFORCE is

$$J_R = \sum_{t=1}^{T} [\log G(\hat{y}_t|X, \hat{Y}_{1:t-1})] \cdot r(t) \quad (2)$$

Instead of calculating the reward using the likelihood, most works rely on a discriminator (Yu et al., 2017; Li et al., 2018a; Liu et al., 2018; Guo et al., 2018) or an evaluator (Li et al., 2018b).

In REINFORCE, the discriminator/evaluator D outputs a single scalar for a whole sentence. D learns using the cross-entropy of the dataset distribution Y and the generated distribution $\hat{Y}$. The objective function J_D of the discriminator/evaluator is

$$J_D = -\log D(X, Y) - \log(1 - D(X, \hat{Y})) \quad (3)$$

So, the reward function for a complete sequence is $r(T) = D(X, \hat{Y}_{1:T})$. However, the discriminator is trained with only completed sequences. REINFORCE establishes the use of Monte-Carlo search with roll-out to sample possible sequences starting from incomplete ones. We define the Monte-Carlo search MC as

$$MC^G(\hat{Y}_{1:t}; N) = \left\{\hat{Y}^1_{1:T}, ..., \hat{Y}^N_{1:T}\right\} \quad (4)$$

In that way, the reward function for all timesteps is

$$r(t) = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} D(X, \hat{Y}^n_{1:T}), \hat{Y}^n_{1:T} \in MC^G(\hat{Y}_{1:t}; N) & \text{for } t < T \\ D(X, \hat{Y}_{1:T}) & \text{for } t = T \end{cases} \quad (5)$$

3 Methodology

Our adversarial setup is composed of two neural networks like previous works: a generator and a discriminator. We make some observations of prior approaches to propose our model. On the one hand, MLE could suffer exposure bias when switching from train to inference. On the other hand, we observe that the REINFORCE algorithm is very fluctuating when training because it relies only on the discriminator. Furthermore, it is slow to train due to the Monte-Carlo search.

We propose a weighted conditional maximum likelihood objective to train our generator. In consequence, we take advantage of the conditional MLE principle and, also, we can guide the training using a penalization function. The penalization function is similar to the reward in REINFORCE. The conditional MLE part let us reduce the variance of training. In addition, we substitute the Monte-Carlo search by using a discriminator that outputs a score for each token.

3.1 Model Architecture

We first define two sequences of tokens $X_{1:T} = \{x_1, ..., x_T\}, Y_{1:T} = \{y_1, ..., y_T\}$ of length T that represent a paraphrase. Let G_θ and D_ϕ be a θ-parameterized generator and a ϕ-parameterized discriminator.

Given X we train G_θ to produce a sequence of tokens $\hat{Y}_{1:T} = (\hat{y}_1, ..., \hat{y}_T)$ that is similar to Y. Given X we train D_ϕ to distinguish between Y and $\hat{Y}$.

In the following sections we also call X, Y, and $\hat{Y}$ as the *condition sentence*, *target sentence*, and *generated sentence* respectively.

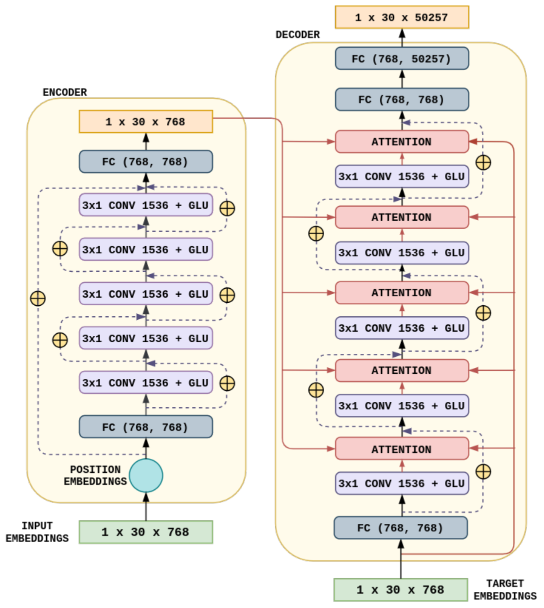

Figure 1: Architecture of generator network

3.1.1 Generator (G_θ)

Our generator is a Convolutional Sequence to Sequence (ConvS2S) model (Gehring et al., 2017). We choose this architecture over a Seq2Seq (Sutskever et al., 2014) and Transformer (Vaswani et al., 2017) because the ConvS2S needs fewer number of parameters to achieve similar results. That let us train our framework using large batch sizes to reduce the generator variance (de Masson d'Autume et al., 2019). Furthermore, the model performs parallel convolutions to speed up the training time. That feature allows us to conduct more experiments.

Figure 1 shows the overall architecture of G_θ. We feed G_θ encoder with the condition sentence embeddings (input embeddings). We add position embeddings on the encoder side. The first encoder layer is a fully connected. Then, each following layer performs iteratively: (a) one-dimensional convolutions without padding and (b) a gated linear unit over the previous layer result. We also add residual connections between non-adjacent layers.

The decoder has convolutional and attention layers interleaved. However, the decoder performs temporal convolutions to avoid leakage of future information. We achieve that by padding the input vector on the left side. There are no position embeddings on the decoder side because they produce a negative impact on the generation. At training time, we feed the decoder with the target sentence to perform convolutions in parallel. At inference

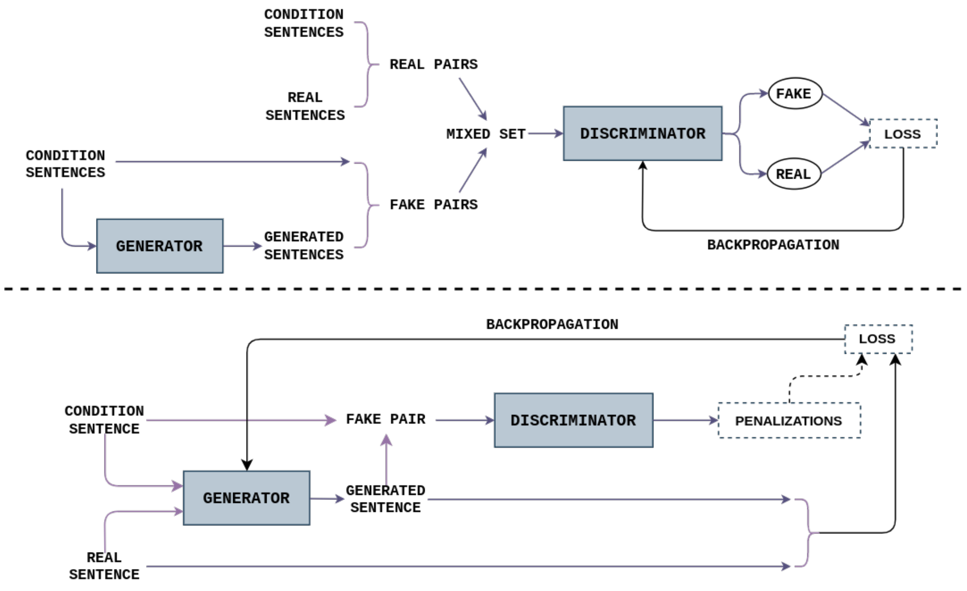

Figure 2: Our model training. Up: Discriminator process. Down: Generator process

time, the decoder is sequential. First, we feed the decoder with our initial token. Then, we repeat two procedures until G_θ generates all words: the decoder performs a forward pass and output a new token; we concatenate the new token to the previous input in order to feed the decoder. Finally, the result vector passes through two dense layers to output the vocabulary probabilities.

3.1.2 Discriminator (D_ϕ)

The architecture of the discriminator is similar to that of the generator. We change the last fully connected layer of D_ϕ decoder to output one number. So, D_ϕ output one score per token. Then, we pass the result to a sigmoid layer that outputs the probability that the tokens belong to the *fake* category.

We feed the encoder with the *condition* sentence, and the decoder with either the *generated* or *target* sentence.

3.2 Training

The training process of D_ϕ and G_θ are different. Figure 2 shows the overall training procedure.

We first generate paraphrases for all condition sentences. We build a mixed set of sentence pairs using the *condition-target (real pairs)* and *condition-generated (fake pairs)*. In that way, we feed the discriminator with a pair of sentences. D_ϕ outputs a score for each generated / target word. So, each pair of sentences is classified as a set of zeros and ones in the *real* or *fake* case, respectively.

D_ϕ learns using the following function

$$J(\phi) = -\log D_\phi(X,Y) - \log(1 - D_\phi(X,\hat{Y})) \quad (6)$$

We train G_θ using a unified learning objective. We multiply the negative log-likelihood loss of each word by the result of our penalization function. The objective function $J(\theta)$ of G_θ is

$$J(\theta) = -\sum_{t=1}^{T} \mathbb{E}_{\hat{Y}_{1:T}\sim Y_{1:T}}[\sum_{\hat{y}_t\in\hat{Y}} \log G_\theta(\hat{y}_t|\hat{Y}_{1:T}, X)\cdot P_{D_\phi}^{G_\theta}] \quad (7)$$

We calculate the G_θ log-likelihood loss using the $real$ sentence as decoder input. Nevertheless, we estimate the penalization function $P_{D_\phi}^{G_\theta}(t)$ with the decoding inference result. Thus, we increase the loss value of tokens that tend to yield infeasible paraphrases.

$P_{D_\phi}^{G_\theta}(t)$ is the discriminator output multiplied by a constant k. The discriminator outputs scores in the interval $[0,1]$ according to the probability that a token is classified as *fake*. That is, tokens classified as fake receive higher penalizations.

$$P_{D_\phi}^{G_\theta}(t) = k\cdot D_\phi(X,\hat{Y}_{1:T}) \quad (8)$$

We avoid the Monte-Carlo search using our discriminator score per word. We update D_ϕ at each training round to improve the quality of our generated sentences.

Algorithm 1 presents the overall procedure to train our model. As first step, we pre-train G_θ using conditional maximum likelihood with the *condition* and *target* samples. Also we pre-train D_ϕ using supervised learning using pairs composed of *condition-real* or *condition-generated*. Then, we start the adversarial training phase for several rounds. First, we sample and calculate $P_{D_\phi}^{G_\theta}$ to train G_θ using equation 7. After updating the parameters, we output a *generated* sample per *condition* sentence using G_θ. That results in a balanced set of fake and real pairs to feed D_ϕ. Finally, we train D_ϕ with Equation 6.

Algorithm 1: Training of the model

```
Result: Trained $G_\theta$
Pre-train $G_\theta$.
Generate samples using $G_\theta$.
Pre-train $D\phi$ with fake and real pairs.
for $n$ rounds do
    for examples do
        Generate a sequence using $G_\theta$.
        Calculate $P_{D_\phi}^{G_\theta}$.
        Train $G_\theta$ using 7
    end
    Generate samples using $G_\theta$.
    for examples$*2$ do
        Train $D_\phi$ using 6
    end
end
```

4 Experiments

In this section, we evaluate our model and compare it with prior methods. We describe the dataset used, experimental setup, baseline methods, and results of our experiments.

4.1 Dataset

To test our model, we used the Quora question pairs dataset. This dataset contains paired questions associated with a label. The pairs are labeled with *1* whether they express the same idea and *0* otherwise. For this work, we decided to use only the duplicated ones. There are around *155K* duplicated questions. We observed that some questions appear in more than one pair.

We built three sets: Quora I, Quora II, and Quora III to evaluate our framework. In Quora I, we randomly selected *133K* pairs: *100K* for training, *30K* for testing, and *3K* for validation. In Quora III, we sampled *83K* pairs: *50K* for training, *30K* for testing, and *3K* for validation. In Quora II, we sampled *30K* pairs for testing which questions do not appear in the training (*50K*) and validation (*3K*) sets. That makes Quora II the most challenging set. It is worth to notice that there are some overlaps of input questions between the training and testing sets in Quora I and III. Table 1 shows the distribution of quantities of our sets.

Dataset	**Train**	**Test**	**Validation**
Quora I	100K	30K	3K
Quora II	50K	30K (Unique)	3K
Quora III	50K	30K	3K

Table 1: Dataset distribution.

4.2 Experimental setup

G and D are 5-layer ConvS2S in the encoder and decoder side. The value of k in the penalization function is 2. For all models, we use the negative log-likelihood as the loss function. We adopted the optimization algorithm Adam (Kingma and Ba, 2014) for pre-training the generator and to train the discriminator. We use $1e-4$ as the learning rate for G_θ and $1e-6$ for D_ϕ without modifying the default betas. Model parameters have been initialized using uniform distribution values, as described in (He et al., 2015). We pre-trained the generator and discriminator for 20 epochs. In the adversarial training phase, we changed the learning rate to $2e-4$ performing 40 rounds of adversarial training. In Monte-Carlo and REINFORCE baselines, a roll-out of size four has been used. The batch size to feed our generator and discriminator is *250*. All samples were generated using greedy decoding. We tuned our hyperparameters manually. We run our experiments in a PC with Intel 9900K and Nvidia Titan RTX for two hours on average for each model. The framework used for implementation has been Pytorch 1.3 (Paszke et al., 2019).

4.3 Input representations

The input representation for deep learning models is important because it is their unique information before solving a task. We tested the influence of the input representations in our results by changing the source of the embedding. In our experiments, we consider three recent methods: The Byte-level BPE extracted from the OpenAI GPT-2 (Radford et al., 2019) with 50257 tokens, the pre-trained wordpiece embeddings from BERT proposed by (Devlin et al., 2019) with 30522 tokens, and the 1 million FastText embeddings trained on 16 billion tokens (Mikolov et al., 2018) considering the

Target Sentence	GPT-2 embeddings	BERT embeddings	Fasttext embeddings
who is best indian hacker?	who are the best hackers in india?	who is the best seo company in india?	who are the best [UNK] in india ?
how can we earn from youtube?	how can i make money from youtube?	how can i make money on youtube?	how can we make money from gmail ?
how can i grow tall?	how can i grow taller?	how can i become taller?	how do i become tall ?
which is best tv series you have seen? and way?	what are the best tv series you have watched?	what are the best tv movies in all time?	what are the best [UNK] movies you have seen ?

Table 2: Generated Sentences by changing the input representation.

100000 most common tokens in their vocabulary. We used ConvS2S architecture in all cases. We only change the input to lowercase and truncate in 30 tokens as preprocessing. Table 3 shows the BLEU-2 score of the testing sets. Table 2 presents some generated sentences using different input representations.

Input Representation	BLEU-2		
	Quora I	Quora II	Quora III
GPT-2 embeddings	**44.84**	**33.48**	**42.48**
BERT embeddings	39.03	19.18	29.28
Fasttext embeddings	31.20	17.37	28.19

Table 3: Comparative results by changing the input representation.

As can be seen in Table 3, the ConvS2S architecture that uses the Byte level BPE embeddings surpasses the BERT embeddings in 11.13 points in the BLEU score in average, and in 14.68 to the FastText. Also, the generated texts of Table 2 presents a correlation with the BLEU scores. The results confirm the superiority of embeddings extracted from contextualizing environments against traditional embeddings. Overall, the presented results indicate that the byte-level BPE embeddings from GPT-2 are the most suitable input representation for our framework.

4.4 Automatic evaluation

We used some automatic metrics to evaluate our framework and compare it with other methods. We use BLEU (Papineni et al., 2002) which evaluates similarities between n-grams, ROUGE (Lin, 2004) which is a common metric in text summarization, METEOR (Denkowski and Lavie, 2014) that considers synonyms, and iBleu (Sun and Zhou, 2012) which penalizes similarities with the source sentence (parroting), as Li et al. (2019); Mao and Lee (2019); Qian et al. (2019) suggest. We consider the following methods as baselines of our research. In all cases, we extracted directly the results reported from their publications.

VAE-SVG and VAE-SVG-eq from Gupta et al. (2018): A variational autoencoder and its modification with fewer parameters. RbM-SL and RbM-IRL from Li et al. (2018b): Reinforcement learning method with an evaluator trained with supervised learning and another with inverse reinforcement learning. DNPG from Li et al. (2019): Multi-granularity encoder and decoder framework using multi-head attention. GAP from Yang et al. (2019): Generative model using REINFORCE with two losses per word. TranSEQ from Egonmwan and Chali (2019): Model with a transformer encoder and an RNN decoder. Our implementation of ConvS2S from Gehring et al. (2017) using byte-level BPE. Our implementation of Transformer from Vaswani et al. (2017) using byte-level BPE as initial embeddings. Our model setup using REINFORCE and another with only Monte-Carlo search.

We compared our results using similar configurations of prior works to make a fair comparison. Quora I and III are identical to the sets of Gupta et al. (2018); Yang et al. (2019); Egonmwan and Chali (2019). Quora I and II are analogous to the sets proposed by Li et al. (2018b). Also, the Quora I set is similar to the set of Li et al. (2019).

Table 4, 5, and 6 show the results for Quora I, II, and III corpora respectively. The results presented refer to the scores of testing sets. Conv-Adv-MC refers to the method with Monte-Carlo discriminator and Conv-Adv-S, to our discriminator. The best results in each metric are in **bold**.

The automatic evaluation results show that our models are competitive against the state of the art baselines. Moreover, the Conv-Adv-S has slightly higher scores than previous methods in Quora I and III. However, RBM-SL is still the best method for generating paraphrases that are completely differ-

Proposed Model	Quora I					
	BLEU-2	BLEU-4	iBLEU	ROUGE-1	ROUGE-2	METEOR
VAE-SVG	-	22.50	-	-	-	25.50
VAE-SVG-eq	-	22.90	-	-	-	25.50
RbM-SL	43.54	-	-	**64.39**	38.11	32.84
RbM-IRL	43.09	-	-	64.02	37.72	31.97
DNPG	-	25.03	18.01	63.73	37.75	-
GAP	44.83	-	-	-	-	32.48
TranSEQ	38.75	-	-	-	-	**35.84**
Transformer	42.03	26.56	20.17	57.80	34.25	29.06
ConvS2S	**44.84**	29.44	21.20	61.72	**38.48**	31.28
REINFORCE	43.96	28.87	20.86	60.43	37.53	31.12
Conv-Adv-MC	44.65	**29.48**	21.08	61.18	38.03	31.01
Conv-Adv-S (ours)	**44.84**	29.07	**21.40**	60.34	37.09	31.27

Table 4: Comparative results on Quora I.

Proposed Model	Quora II			
	BLEU-2	ROUGE-1	ROUGE-2	METEOR
RbM-SL	**35.81**	**57.34**	**31.09**	**28.12**
RbM-IRL	34.79	56.86	29.90	26.67
Transformer	25.45	38.57	18.22	17.71
ConvS2S	33.48	48.30	26.83	23.07
REINFORCE	32.41	47.58	26.19	22.15
Conv-Adv-MC	32.73	47.76	26.63	22.58
Conv-Adv-S	33.27	47.73	26.25	23.03

Table 5: Comparative results on Quora II.

Proposed Model	Quora III		
	BLEU-2	BLEU-4	METEOR
VAE-SVG	-	17.1	22.20
VAE-SVG-eq	-	17.4	22.20
GAP	37.18	-	22.24
TranSEQ	38.75	-	**33.73**
Transformer	35.74	16.48	24.73
ConvS2S	42.48	27.02	29.52
REINFORCE	42.05	26.73	29.20
Conv-Adv-MC	41.98	26.84	29.45
Conv-Adv-S	**42.81**	**27.33**	29.70

Table 6: Comparative results on Quora III.

ent than the training set (Quora II).

The scores on BLEU and ROUGE indicate that our model produces paraphrases that are more similar to the targets. Also, the iBLEU scores suggest that our model generates more diverse paraphrases than some of the prior works. However, the METEOR scores indicate that some of the previous baselines use more synonyms when generating paraphrases than our model does.

4.5 Human evaluation

We make a human evaluation of the generated models because we believe that the automatic evaluation is not always accurate. We randomly select 120 condition-target questions and the generated of two methods of the Quora I testing set, and we distribute them to 4 evaluators using a form. We ask the evaluators to verify three main aspects in scores from 1 to 5:

- **Relevance**: Whether the paraphrase has the same meaning of the original sentence and does not lose information.
- **Fluency**: Whether the paraphrase has a correct grammar and use of vocabulary.
- **Diversity**: Whether the paraphrase varies syntactically and semantically.

Table 7 presents the average ratings given by human evaluators. It is worth to notice that all appraisals are from 1 to 5, being five the highest score.

Model	Relevance	Fluency	Diversity
Reference	3.97	4.42	3.22
ConvS2S	3.98	4.32	2.65
Conv-Adv-S (ours)	3.97	4.24	2.77

Table 7: Results of human evaluation.

From the results, we analyze each evaluated aspect: The three paraphrases have equivalent Relevance scores. We infer that this is due to some reference examples that lose some extra information in the paraphrase and for the random sampling. Overall, the scores indicate that our model can produce paraphrases that are highly related to the original input questions.

Although the ConvS2S model has a better fluency than our method with non-statistical significant differences (paired t-test, p-value$\sim$0.21), both are near the reference examples. We believe that the process of correcting some specific words could cause disorder when decoding some sentences. Besides, we assume that another decoding algorithm

Input Sentence	ConvS2S	Conv-Adv-S (ours)	Target sentence
what is the difference between militants and terrorists?	🟨 what is the difference between terrorists and terrorists?	🟨 what is the difference between a person and terrorists?	🟩 what is the difference between terrorists and militants?
who is going to win, trump or hillary?	🟩 who will win the election, trump or clinton?	🟩 who will win, trump or clinton?	🟩 who will win, trump or clinton?
is it possible to advertise on quora?	🟥 is it possible to advertise on quora?	🟩 is promotion allowed on quora?	🟩 can we advertise our business on quora?
how do you train your memory to memorize things fast?	🟨 how do i memorize my memory?	🟩 how can i memorize things faster?	🟩 how can i memorize things faster?
what are the top 5 mobile app development companies in india?	🟥 what are the top 5 mobile app development companies in india?	🟩 which is the best mobile app development company in india?	🟩 what is the best mobile app development company in india?
how can i persuade my parents to let me wear makeup?	🟩 how do i convince my parents to let me wear makeup?	🟨 how do i let my parents to let me wear makeup?	🟩 how do i convince my parents to let me wear makeup?
why doesn't germany pursues indigenous jet engine development?	🟥 why doesn't germany angulic in the world?	🟥 why doesn't the germany always a bit of it's limited/2.2.2.50 & sandys have been fired in the	🟩 why doesn't germany produce jet engines?

Table 8: Generated sentences on Quora I testing set.

could benefit the generation of fluent paraphrases.

Our model produces more diverse sentences than ConvS2S does. However, the difference is not statistical significant (p-value∼0.19). Also, the reference sentences are still more diverse than the outputs of our model. We believe that a penalization function for similarity with the source sentence could help to improve the results.

Overall, human evaluation shows that our model produces relevant and fluent paraphrases, which indicates that our model works properly. Although we improved the diversity factor of the ConvS2S, it is still non-comparable with the variety of the original distribution.

As a case study, Table 8 presents some of the sentences that our model and the ConvS2S generated in the test set of Quora I. We color the cells depending on our criterion of the quality of each paraphrase. The red color represents a bad paraphrase (a repetition from the source question, a nonsense sentence, or a not related question). The yellow color represents a paraphrase with some missing information. The green color represents a correct paraphrase.

4.6 Discussion

The presented results on the experimentation of the representations of the input show that the embeddings of pre-trained models are better input representations than the ones provided by classic algorithms of word-embeddings. The results concur with the study of (Ethayarajh, 2019). Furthermore, we found that the input representation has a high impact on the framework results (difference up to 14.68). Also, the results indicate that the GPT-2 has more robust embeddings in the first layer than the ones provided by BERT. We infer that BERT relies more on its contextual similarities calculated in its higher layers than GPT-2 does, as is suggested in previous studies (Ethayarajh, 2019; Hoover et al., 2019).

The comparison with previous baselines indicates that our framework achieves competitive results against the state-of-the-art methods considering all automatic metrics. Further, we improved some benchmarks on BLEU, ROUGE, and METEOR. The results indicate that the weighted adversarial loss is a suitable option to REINFORCE, and it provides generation diversity to our ConvS2S implementation (improvement of 0.12 in human evaluation). The BLEU and ROUGE scores indicate that our model has similarities with the target questions. However, it also has a lack of diversity as the METEOR score shows. Similar to (Qian et al., 2019; Banerjee and Lavie, 2005), the results from the automatic evaluation concur with the human evaluation, especially on diversity.

5 Conclusions

We propose an adversarial setup to address the paraphrase generation task. The automatic evaluation results show some improvements over previous baselines. The human evaluation suggests a tradeoff between the fluency and diversity of the generated paraphrases of the fine-tuned model. We conclude that our setup is a suitable option to RE-

INFORCE and MLE training. In addition, the case study shows that our method helps to improve the quality of paraphrases in general.

Acknowledgments

This work was supported by grant 234-2015-FONDECYT (Master Program) from Cienciactiva of the National Council for Science, Technology and Technological Innovation (CONCYTEC-PERU).

References

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

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *Advances in Neural Information Processing Systems*, pages 1171–1179.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984.

Claude Coulombe. 2018. Text data augmentation made simple by leveraging nlp cloud apis. *arXiv preprint arXiv:1812.04718*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the EACL 2014 Workshop on Statistical Machine Translation*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 875–886.

Elozino Egonmwan and Yllias Chali. 2019. Transformer and seq2seq model for paraphrase generation. In *Proceedings of the 3rd Workshop on Neural Generation and Translation*, pages 249–255.

Kawin Ethayarajh. 2019. How contextual are contextualized word representations? comparing the geometry of bert, elmo, and gpt-2 embeddings. 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)*, pages 55–65.

William Fedus, Ian Goodfellow, and Andrew M Dai. 2018. Maskgan: Better text generation via filling in the _. *arXiv preprint arXiv:1801.07736*.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. 2018. Long text generation via adversarial training with leaked information. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

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

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *Proceedings of the IEEE international conference on computer vision*, pages 1026–1034.

Tianxing He, Jingzhao Zhang, Zhiming Zhou, and James Glass. 2019. Quantifying exposure bias for neural language generation. *arXiv preprint arXiv:1905.10617*.

Benjamin Hoover, Hendrik Strobelt, and Sebastian Gehrmann. 2019. exbert: A visual analysis tool to explore learned representations in transformers models. *arXiv preprint arXiv:1910.05276*.

Shaohan Huang, Yu Wu, Furu Wei, and Ming Zhou. 2018. Dictionary-guided editing networks for paraphrase generation. *arXiv preprint arXiv:1806.08077*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North*

American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), volume 1, pages 1875–1885.

Xiaoqi Jiao, Yichun Yin, Lifeng Shang, Xin Jiang, Xiao Chen, Linlin Li, Fang Wang, and Qun Liu. 2019. Tinybert: Distilling bert for natural language understanding. *arXiv preprint arXiv:1909.10351*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yang Li, Quan Pan, Suhang Wang, Tao Yang, and Erik Cambria. 2018a. A generative model for category text generation. *Information Sciences*, 450:301–315.

Yitong Li, Trevor Cohn, and Timothy Baldwin. 2017. Robust training under linguistic adversity. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 21–27.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018b. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Xinyue Liu, Xiangnan Kong, Lei Liu, and Kuorong Chiang. 2018. Treegan: Syntax-aware sequence generation with generative adversarial networks. *arXiv preprint arXiv:1808.07582*.

Sidi Lu, Lantao Yu, Siyuan Feng, Yaoming Zhu, and Weinan Zhang. 2019. Cot: Cooperative training for generative modeling of discrete data. In *International Conference on Machine Learning*, pages 4164–4172.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018. Query and output: Generating words by querying distributed word representations for paraphrase generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 196–206.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 1, Long Papers*, volume 1, pages 881–893.

Hong-Ren Mao and Hung-Yi Lee. 2019. Polly want a cracker: Analyzing performance of parroting on paraphrase generation datasets. 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)*, pages 5962–5970.

Cyprien de Masson d'Autume, Shakir Mohamed, Mihaela Rosca, and Jack Rae. 2019. Training language gans from scratch. In *Advances in Neural Information Processing Systems*, pages 4302–4313.

Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. 2018. Advances in pre-training distributed word representations. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pages 3111–3119.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting on association for computational linguistics*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. Pytorch: An imperative style, high-performance deep learning library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

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

Lihua Qian, Lin Qiu, Weinan Zhang, Xin Jiang, and Yong Yu. 2019. Exploring diverse expressions 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)*, pages 3164–3173.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI Blog*, 1(8):9.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2015. Sequence level training with recurrent neural networks. *arXiv preprint arXiv:1511.06732*.

Suranjana Samanta and Sameep Mehta. 2017. Towards crafting text adversarial samples. *arXiv preprint arXiv:1707.02812*.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual smt system for paraphrase generation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Short Papers-Volume 2*, pages 38–42. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7176–7183.

Ronald J Williams. 1992. Simple statistical gradient-following algorithms for connectionist reinforcement learning. *Machine learning*, 8(3-4):229–256.

Ronald J Williams and David Zipser. 1989. A learning algorithm for continually running fully recurrent neural networks. *Neural computation*, 1(2):270–280.

Qizhe Xie, Zihang Dai, Eduard Hovy, Minh-Thang Luong, and Quoc V Le. 2019. Unsupervised data augmentation for consistency training.

Qian Yang, Dinghan Shen, Yong Cheng, Wenlin Wang, Guoyin Wang, Lawrence Carin, et al. 2019. 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)*, pages 3123–3133.

Lantao Yu, Weinan Zhang, Jun Wang, and Yong Yu. 2017. Seqgan: Sequence generative adversarial nets with policy gradient. In *AAAI*, pages 2852–2858.