Exploring Diverse Expressions for Paraphrase Generation

Lihua Qian[1], **Lin Qiu**[1], **Weinan Zhang**[1], **Xin Jiang**[2], **Yong Yu**[1]
[1]Shanghai Jiao Tong University
{qianlihua, lqiu, wnzhang, yyu}@apex.sjtu.edu.cn
[2]Noah's Ark Lab, Huawei Technologies
jiang.xin@huawei.com

Abstract

Paraphrasing plays an important role in various natural language processing (NLP) tasks, such as question answering, information retrieval and sentence simplification. Recently, neural generative models have shown promising results in paraphrase generation. However, prior work mainly focused on single paraphrase generation, while ignoring the fact that diversity is essential for enhancing generalization capability and robustness of downstream applications. Few works have been done to solve diverse paraphrase generation. In this paper, we propose a novel approach with two discriminators and multiple generators to generate a variety of different paraphrases. A reinforcement learning algorithm is applied to train our model. Our experiments on two real-world datasets demonstrate that our model not only gains a significant increase in diversity but also improves generation quality over several state-of-the-art baselines.

1 Introduction

Paraphrases refer to texts with identical meaning but expressed in different ways. Paraphrase generation has a wide range of applications. For example, paraphrases can be used to extend the reference texts for the automatic evaluation of text generation tasks, which make the evaluation more robust and accurate (Madnani and Dorr, 2010). In addition to increasing reference texts, paraphrases can also augment training data for text classification problems, especially for tasks lack of training data. Besides, paraphrases can be used as variations of queries for information retrieval and question answering systems. Query paraphrasing can narrow the gap between system comprehension and user questions (Fader et al., 2014; Yin et al., 2015). And systems can obtain better results from multiple feedbacks for different query variants.

Traditional paraphrase generation approaches leverage manual-crafted rules (McKeown, 1983; Hassan et al., 2007). In recent years, sequence to sequence learning techniques have achieved improvements in a variety of NLP tasks, such as machine translation (Bahdanau et al., 2014; Cho et al., 2014), text summarization (Nallapati et al., 2016) and dialogue systems (Serban et al., 2017). Specifically, the paraphrase generation can also be formulated as a sequence to sequence problem. Existing neural paraphrase generation works (Prakash et al., 2016; Gupta et al., 2018; Li et al., 2018) also demonstrate the effectiveness of deep learning method.

Although prior paraphrase generation methods mainly consider only one single output for each input, the nature of paraphrasing indicates that we can paraphrase one sentence into several different sentences. Therefore, we hope to generate a variety of paraphrases while ensuring quality. However, most of the previous work dedicated to improving the quality of top generated sentence, but did not conduct in-depth research on the diversity of paraphrase generation. In fact, diversity is very important in many applications. More diverse paraphrases will be beneficial to robustness and accuracy of automatic text evaluation, text classification, and can avoid the blandness caused by repetitive patterns.

A straightforward way to generate multiple different sentences is using beam search to select the top k sentences. However, beam search generates suboptimal sentences, and the resulting sentences are very similar to each other. Gupta et al. (2018) employ a variational auto-encoder framework to produce multiple sentences according to different noise input. Xu et al. (2018b) learn a shared decoder with different decoder embeddings for various decoding pattern. Although we can get multiple outputs with different noise inputs or decoder embeddings, different outputs may have slight changes or even no difference.

In order to address the problem, in this paper, we utilize multiple generators to generate diverse paraphrases without sacrificing quality. Reinforcement learning makes it possible for every generator to explore different generation pattern, breaking the restriction of learning the same target sentences given by data in supervised learning. Thus, we design a reinforcement learning algorithm to guide the models learning procedure. The proposed framework contains two discriminators. One discriminator is to examine whether two sentences convey the same meaning, the other one is to distinguish the sentence is generated by which generator. The higher the confidence of one text be judged as generated by a certain generator is, indicating the more obvious the differences between that text and texts generated by other generators are. We combine the scores of the two discriminators as guiding signals to the generators using reinforcement learning. Therefore, generators can learn to synthesize texts with apparent differences and good quality.

Our approach can automatically discover the solution to diverse paraphrase generation through the jointly learning of the discriminators and generators. And the model uses the same data pairs as in learning mono-paraphrase generation without extra annotations. Experiments on two real-world datasets show that our model improves both in quality and diversity. There are mainly two contributions in our work: the first one is proposing a novel framework as well as a corresponding learning procedure to generate various paraphrases, the second one is performing experiments demonstrating that our method achieves improvement regarding both generation diversity and quality.

2 Related Work

Paraphrase generation In the early years, various traditional techniques have been developed to solve the paraphrase generation problem. McKeown (1983) makes use of manually defined rules. Hassan et al. (2007) use thesaurus-based method for lexical substitution. And statistical machine translation (SMT) has also been used. Quirk et al. (2004) train SMT tools on a large number of sentence pairs collected from newspapers. Wubben et al. (2010) propose a phrase-based SMT model trained on aligned news headlines. Zhao et al. (2008) utilize multiple resources to strengthen a log-linear SMT model. Recently, deep neural models have also been applied to paraphrase generation due to their great success on natural language processing tasks. Prakash et al. (2016) design a deep stacked network with residual connections, Gupta et al. (2018) propose a conditional variational auto-encoder which can produce multiple paraphrases, Iyyer et al. (2018) learn a model to generate syntactically controlled paraphrase, Li et al. (2018) propose a generator-evaluator architecture coupled with deep reinforcement learning. Although being similar to the architecture proposed in (Li et al., 2018), our framework targets a different goal. Our work focuses on generating multiple diverse paraphrases, while theirs dedicates to improving the quality of the top generated paraphrase. The main difference is that our approach utilizes a generator discriminator to encourage more diverse paraphrases, which is essential for diversity.

Diverse text generation A few works have explored to produce diverse generation by changing decoding schemes or introducing random noise. Methods that change decoding schemes are orthogonal and complementary to our work. Li et al. (2016) modify the score function in decoding to encourage usage of novel words and penalize words after the same partially generated sentence. Dai et al. (2017) utilize conditional generative adversarial network (GAN) to generate diverse image caption according to the input noise. Gupta et al. (2018) employ a variational auto-encoder framework with both encoder and decoder conditioned on source input. Shi et al. (2018) employ inverse reinforcement learning for unconditional diverse text generation. Xu et al. (2018a) propose a modified GAN to generate diverse and informative outputs for different inputs. Xu et al. (2018b) train a generator with different embeddings to generate multiple paraphrases, only the decoder embedding with lowest cross entropy can get updated.

3 Methods

3.1 Problem Definition

Given a source sentence X, we aim to generate multiple target sentences $S = \{Y_1, Y_2, ..., Y_k\}$. Specifically, our goal is to learn a model which can generate S such that every Y_i (i=1..k) has the same meaning as X and obvious differences between each other.

3.2 Models

Our diverse paraphrase generation framework consists of three parts: one paraphrase discriminator, one generator discriminator and multiple generators. Paraphrase discriminator determines whether two sentences have the same meaning, and the generator discriminator determines the sentence is generated by which generator. The goal of multiple paraphrase generators is to generate sentences that are judged as paraphrases of the input sentence, and the differences between sentences should be as large as possible. The intuition is that sentences with quite different expressions can be easily distinguished by the generator discriminator and will be judged as generated by the corresponding generator with high confidence.

3.2.1 Paraphrase generator

We frame paraphrase generation as a sequence-to-sequence (Seq2Seq) problem and adopt the encoder-decoder framework. Multiple generators share the same encoder yet with their own decoder. The generation posterior of the i-th generator is:

$$G_i(Y|X) = \prod_{t=1}^{T} P(y_t|y_{1:t-1}, X; \theta_i),$$

where $y_{1:t-1}$ indicates the subsequence from timestep 1 to timestep $t-1$, T denotes the length of output sentence Y, θ_i represents the parameter of the i-th generator, we omit θ_i in the following illustration for conciseness. We use pointer-generator (See et al., 2017) for our generators to alleviate the out-of-vocabulary problem and enhance the performance. In pointer-generator, the decoder can generate not only words from the chosen vocabulary as standard decoders but also words from the input sentences by copying them from the input. The probability of copying words can be represented as

$$p_{copy}(y_t|y_{1:t-1}, X) = a_{ti},$$

$$a_{ti} = \frac{exp(f(s_{t-1}, h_i))}{\sum_i exp(f(s_{t-1}, h_i))},$$

where s_{t-1} is the decoder state at timestep $t-1$, h_i is the encoder hidden state at timestep i. p_{copy} is actually a_{ti}, which is the attention distribution at timestep t. We use attention in (Bahdanau et al., 2014), thus f is a linear function. The probability of extracting the next word from vocabulary is:

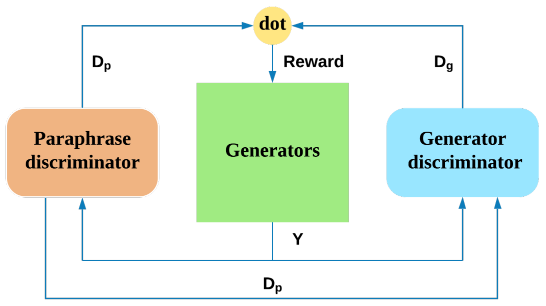

Figure 1: The overall framework

$$p_{vocab}(y_t|y_{1:t-1}, X) = g(s_{t-1}, y_{t-1}, c_t),$$

$$c_t = \sum_i a_{ti} * h_i,$$

where y_{t-1} is the word generated last step, c_t means the context vector at timestep t, g is a linear function with softmax output. The overall probability of the next word is

$$p(y_t|y_{1:t-1}, X) = p_{gen} * p_{vocab} + (1 - p_{gen}) * p_{copy},$$

$$p_{gen} = m(s_t, y_{t-1}, c_t)$$

where m is a linear transformation with sigmoid activation output. p_{gen} acts as a gate to control whether the next word is generated from vocabulary or is copied from the input.

3.2.2 Paraphrase discriminator

We follow (Li et al., 2018) to cast paraphrase recognition as a sentence matching problem. In our work, we use a model with two recurrent neural networks (RNN). Assume giving a pair of sentences (X, Y), paraphrase discriminator will score a value to evaluate the semantic similarity between X and Y.

$$\alpha = RNN_\alpha(X),\ \beta = RNN_\beta(Y),$$

$$\gamma = \tanh(W_\alpha * \alpha + b_\alpha) * \tanh(W_\beta * \beta + b_\beta),$$

$$D_p = softmax(W_p * f_c(\gamma) + b_p),$$

sentence X and sentence Y are encoded using RNN-based encoder. The encoded representations α and β are integrated into γ. Then we input γ into a fully connected network f_c and finally output a softmax classification probability, which represents the degree of meaning similarity between

X and Y. We optimize the cross entropy loss for paraphrase discriminator.

$$L = -\log(D_p(X, Y+)) - \log(1 - D_p(X, Y-)),$$

where$(X, Y-)$ means sentence $Y-$ is not a paraphrase of X and $(X, Y+)$ means sentence $Y+$ is a paraphrase of X. Paraphrase discriminator separate the negative sentence pairs from the positive ones.

3.2.3 Generator discriminator

In order to distinguish sentences generated by different generators, we learn a sentence discriminator. Suppose there is a set of sentences $\{S_1, S_2, ..., S_k\}$, where S_i denotes sentences generated by the i-th generator. We hope our generator discriminator can separate sentences in S_i $(i = 1..k)$ from sentences generated by other generators. If there are obvious differences between sentences in S_i and those generated by other generators, generator discriminator can determine the sentence is generated by the i-th generator with high confidence. Otherwise, if sentences in S_i are very similar to those generated by other generators, generator discriminator will get confused and misclassify the sentences into another generator, thus confidence will be low as well. For simplicity, we use a RNN to extract feature from sentences and use a softmax output as activation function.

$$e = RNN(Y),\ D_g(y|Y) = softmax(e)$$

where e indicates the sentence feature and y means which generator generates Y. The generator discriminator is pretrained using data generated by the pretrained generators. During the reinforcement learning phase, generator discriminator updates iteratively using new samples produced by all generators.

3.3 Learning

3.3.1 Maximum likelihood estimation

The most typical technique to train a Seq2Seq model is teacher forcing (Williams and Zipser, 1989), which is also referred to as maximum likelihood estimation (MLE).

$$L = \sum_{t=1}^{T} \log p(y_t^* | y_{1:t-1}^*, X) \qquad (1)$$

where y_t^* denotes the t-th word in the target sentence given by training sample. We pretrain all generators using teacher forcing. However, teacher forcing suffers from exposure bias (Ranzato et al., 2015). During training, the next word to generate is conditioned on the correctly annotated word subsequence. But in the inference stage, the model predicts the next word conditioned on words generated by itself, which causes inconsistency between training and inference. As model may generate wrong words, future generation conditioned on the generated subsequence will also be affected, leading to accumulated errors. One solution to this problem is finetuning generators via reinforcement learning after pretraining.

Algorithm 1: Training Procedure of our framework

```
Input  : A data with paraphrase pairs $\{(X, Y)\}$ and
         non-paraphrase pairs $\{(X, Y-)\}$
Output : Generators $\{G_i\}_{i=1}^{i=k}$
1  Random initialize all generators $\{G_i\}_{i=1}^{i=k}$;
2  Pre-train all generators $G_i$ with $\{(X, Y)\}$;
3  Generate sentences $\{S_i\}_{i=1}^{i=k}$ , $S_i = \{\hat{Y}\}_i$ using $G_i$;
4  Train paraphrase discriminator with $\{(X, Y)\}$ and
   $\{(X, Y-)\}$;
5  Pre-train generator discriminator with $\{S_i\}_{i=1}^{i=k}$;
6  while not converge do
7      Sample sentences $\{X\}$ from the paraphrase corpus;
8      for $i = 1$ To $k$ do
9          Using $G_i$ to generate sentences $\{\hat{Y}\}_i$ by
           sampling and $\{\bar{Y}\}_i$ by greedy search;
10         Compute the rewards using Eq(3);
11         Update generators by minimizing Eq(4);
12     end
13     Generate sentences $\{S_i\}_{i=1}^{i=k}$ using $\{G_i\}_{i=1}^{i=k}$;
14     Update generator discriminator with $\{S_i\}_{i=1}^{i=k}$ by
       minimizing Eq(7);
15 end
16 Return $\{G_i\}_{i=1}^{i=k}$
```

3.3.2 Reinforcement learning

Regarding a generator as an agent interacting with the environment (i.e. the word input and context vector). The objective function of the generator can be written as

$$L = \sum_{\hat{Y}} R(\hat{Y}), \hat{Y} \sim \pi_\theta \qquad (2)$$

$\hat{Y}$ indicates the word sequence sampled from the generator. R is the reward function, and a batch of sentences is sampled to approximate the objective function since it is intractable to use all sentences. The optimization objective is to minimize the negative expected rewards of generated texts, in other words, maximize the expected rewards.

The reward of each sentence is determined by the paraphrase discriminator and the generator discriminator together. The reward of the i-th generator is

$$R_i = D_p(X, Y_i) \odot D_g(y = i|Y_i), \tag{3}$$

where Y_i indicates the sentence generated by the i-th generator. The i-th generator can get a high reward only when sentence Y_i is recognized as a paraphrase of X by the paraphrase discriminator and the differences between Y_i and sentences generated by other generators is so large that generator discriminator takes Y_i as a sentence generated by the i-th generator.

Specifically, we use the self-critical algorithm (Rennie et al., 2017) as our reinforcement learning method. In order to reduce the variance of policy gradient method, a typical technique is subtracting baseline values from the original rewards. In the self-critical algorithm, the baseline is the reward of sentences generated in inference. At each training iteration, the generator generates two sentences. One sentence is sampled from the output probability distribution, $\hat{y}_t \sim p(y_t|\hat{y}_{1:1-t}, X)$. The other sentence is generated by greedy search, $\bar{y}_t = \arg\max_{y_t} P(y_t|\bar{y}_{1:t-1}, X)$, the word with the highest probability is selected at each step. The reward of the sentence generated by greedy search is used as a baseline. We use rewards subtracted by baseline values to update the model

$$L = \sum_{t=1}^{T} (R(\hat{y}) - R(\bar{y})) \log p(y_t|y_{1:t-1}, X), \tag{4}$$

which means that we compare the rewards of the sampled sentences $\hat{y}$ and those of the sentences $\bar{y}$ which are produced by greedy search. Only the generated sentences that outperform the sentence $\bar{y}$ are given positive signals. As a result, Rewards will be increased as the generators increase the generation probability of better sequences while decreasing the chances of worse sequence generation.

3.3.3 The overall objective

Our goal is to generate multiple diverse paraphrases of high quality. We illustrate the reason why our model can achieve this goal from the aspect of the whole optimization objective. Given input X, we hope to generate multiple high-quality sentences Y. We make the assumption that y is only dependent on Y, that is, $D_g(y|Y, X) = D_g(y|Y)$. The weaker condition forces the generator discriminator to only focus on the generated sentences and encourages stronger diversity. The overall objective can be written as:

$$\begin{aligned}
&\sum_{Y} R(X, Y) P(Y|X) \\
=&\sum_{Y}\sum_{y} R(X, Y) P(Y, y|X) \\
=&\sum_{Y}\sum_{i=1}^{k} R(X, Y) D_g(y = i|Y) G_i(Y|X)
\end{aligned} \tag{5}$$

where $R(X, Y)$ is the reward evaluating output Y for input sentence X. Multiple generators G_i generate their own sequences for each input and D_g learns to separate sentences generated by different generators. We can define different rewards for corresponding applications and we use $D_p(X, Y)$ for our paraphrase task. We define θ_d as the parameter of generator discriminator and θ_i as the parameter of the i-th generator. The derivative of $J(\theta)$ is computed as follow:

$$\begin{aligned}
&\nabla_\theta J(\theta) \\
=&\sum_{i=1}^{k}\sum_{Y} [R(X, Y) G_i(Y|X) \nabla_{\theta_d} D_g(y = i|Y) \\
&\quad + R(X, Y) D_g(y = i|Y) \nabla_{\theta_i} G_i(Y|X)] \\
=&\sum_{i=1}^{k} \mathbb{E}_{G_i(Y|X)} [R(X, Y) \nabla_{\theta_d} D_g(y = i|Y) \\
&\quad + R(X, Y) D_g(y = i|Y) \nabla_{\theta_i} \log(G_i(Y|X))]
\end{aligned} \tag{6}$$

Since the gradient can not be propagated backward to G_i and D_g by gradient descent, we use reinforcement learning to update both D_g and G_i. And directly optimizing the above objective will reduce the reward of a good output Y^* to zero for all generators except the one with the highest generation probability for this output Y^*. We should lower the reward of that good output for other generators, but reducing the reward of Y^* to zero will affect the stability of other generators. To address this issue, we can regularize D_g by adding entropy or we propose to scale the gradient of D_g by $1/D_g$. The derived cost function of generator discriminator is:

$$L = -\sum_{i=1}^{k} \mathbb{E}_{G_i(Y|X)} R(X, Y) \log(D_g(y = i|Y)) \tag{7}$$

The outputs generated by G_i are not equally contributed to $D_g(y = i|Y)$, outputs that have better quality are more encouraged by D_g. High-quality sentences Y generated by G_i maximize $D_g(y = i|Y)$ while minimizing $D_g(y = j|Y)$ $(j \neq i)$ to force other generators to generate different sentences. The i-th generator maximizes $R(X, Y)D_g(y = i|Y)$ so that sentences generated by the i-th generator satisfy both the quality and diversity requirement. The complete learning procedure is shown in algorithm 1.

4 Experiment

4.1 Dataset

	Generator			Paraphrase discriminator	
Dataset	#Train	#Validation	#Test	#Positive	#Negative
Quora	100K	3K	30K	100K	160K
Twitter	110K	1K	5K	10K	40K

Table 1: Statistics of datasets.

To validate our model's capability to generate diverse paraphrases, we conduct experiments on two real-world datasets: Quora question paraphrase dataset [1] and Twitter URL paraphrasing dataset (Lan et al., 2017).

Quora dataset contains a large corpus of question pairs collected from Quora. The Quora dataset totally contains about 400k data pairs. All the data are manually labeled whether two sentences convey the identical meaning. We randomly select 100k pairs for training, 3k for validation and 30k for test. The twitter dataset contains sentence pairs collected from Twitter. It also marks whether the sentence pairs are paraphrases. But the difference is that some of them are manually labeled and some are automatically labeled by the algorithm. For generators, we split test set and validation set from the manually labeled subset. The training set contains the rest data from the manually labeled subset and data from the automatically labeled paraphrase pairs. In order to avoid bias induced by the labelling algorithm, we only use the manually labeled data for the paraphrase discriminator. A summary of dataset statistics is given in Table 1.

4.2 Evaluation metric and baseline

We evaluate our model's performance for both quality and diversity. For sentence quality, we use two commonly used metrics: BLEU (Papineni et al., 2002) and METEOR (Lavie and Agarwal, 2007). BLEU measures word overlapping between generated sentences and reference texts. METEOR introduces stemming and synonym matching to alleviate problems caused by only exact matching in BLEU. BLEU and METEOR scores are averaged for multiple generated sentences. To evaluate sentence quality more reliably, we also conduct human evaluations to compare the quality of sentences generated by different models. For diversity, we use self-BLEU (Zhu et al., 2018). To compute self-BLEU of a set of outputs, every output is picked once as candidate and others are used as references to compute BLEU score. And the average of all the BLEU scores computed for this set is self-BLEU. A lower self-BLEU score indicates better diversity.

Several basic models and existing neural-based methods are used as baseline models for comparison. For both datasets, we report results of attentive Seq2Seq (Bahdanau et al., 2014), variational auto-encoder (VAE-SVG-eq) (Gupta et al., 2018), conditional GAN (Dai et al., 2017) with the generator replaced by Seq2Seq, diversity-promoting GAN (DP-GAN) (Xu et al., 2018a), the generator with inverse reinforcement learning (IRL) (Shi et al., 2018), the generator with multiple decoder embeddings (D-PAGE) (Xu et al., 2018b) and pointer-generator (See et al., 2017). We use beam search for attentive Seq2Seq (Seq2Seq-BS), and pointer-generator (PG-BS) to generate multiple outputs. Top three outputs of each method are obtained for evaluation.

4.3 Implementation

We use the following experimental setting for our model. For generators, we randomly initialize all generators and train them by teacher forcing in the same way. We use three generators and utilize single-layer LSTM (Hochreiter and Schmidhuber, 1997) for both encoder and decoders. The hidden dimension of encoder and decoders is set to 256, the word embedding dimension is 128. We use Adam optimizer (Duchi et al., 2011) with a learning rate of 1e-3 in MLE pretrain. In reinforcement learning, the learning rate decreases to 2e-5 and the batch size is fixed at 32. We also use groundtruth with reward of D_g to stabilize the training. We strengthen the performance of D-PAGE by replacing the basic Seq2Seq with pointer-generator.

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

	Quora (Quality)				Quora (Diversity)		
Models	BLEU2↑	BLEU3↑	BLEU4↑	METEOR↑	self-BLEU2↓	self-BLEU3↓	self-BLEU4↓
Seq2Seq-BS	37.03	27.53	21.11	27.78	73.89	66.51	59.33
VAE-SVG-eq	34.78	25.96	19.97	26.45	89.82	87.53	85.43
GAN	36.22	27.27	21.12	26.63	92.85	91.22	89.76
DP-GAN	36.60	26.92	20.42	28.17	73.76	66.37	59.12
IRL	35.53	26.19	19.94	27.29	76.76	70.22	63.59
D-PAGE	38.70	29.23	**22.66**	28.54	90.10	87.66	85.41
PG-BS	**38.91**	**29.26**	22.54	28.88	75.60	68.72	61.89
ours	36.84	27.46	20.98	**29.28**	**56.38**	**47.76**	**40.55**

Table 2: Performances on Quora dataset.

	Twitter (Quality)				Twitter (Diversity)		
Models	BLEU2↑	BLEU3↑	BLEU4↑	METEOR↑	self-BLEU2↓	self-BLEU3↓	self-BLEU4↓
Seq2Seq-BS	32.69	28.25	24.99	22.51	82.79	80.29	77.98
VAE-SVG-eq	26.43	23.04	20.57	18.34	91.46	90.51	89.67
GAN	23.10	20.45	18.47	15.33	94.35	93.75	93.22
DP-GAN	33.07	28.68	25.49	22.84	82.52	80.05	77.63
IRL	32.96	28.60	25.39	22.72	83.53	81.22	78.95
D-PAGE	32.95	28.82	25.88	22.59	88.35	86.45	84.76
PG-BS	33.86	29.42	26.15	23.52	82.57	79.99	77.48
ours	**34.23**	**29.66**	**26.38**	**24.29**	**65.83**	**61.17**	**57.45**

Table 3: Performances on Twitter dataset.

	Quora		Twitter	
Models	Fluency	Consistency	Fluency	Consistency
D-PAGE	4.21	3.44	3.66	3.08
PG-BS	4.20	3.34	3.85	3.17
DP-GAN	4.27	3.49	4.09	3.30
ours	**4.57**	**3.82**	**4.24**	**3.59**

Table 4: Human evaluation results.

For paraphrase discriminator and generator discriminator, the sentence encoders are all built with one-layer LSTM with 256-dimensional hidden units and the dimension of word embeddings is set to 256. For each of the two discriminators, Adam optimizer with a learning rate of 1e-4 is used to optimize the loss function. The batch size is kept at 64 for paraphrase discriminator. We employ a 3-layer fully connected network with ReLU activation in paraphrase discriminator and set the dropout to 0.2.

5 Results

Results of automatic evaluation on Quora is shown in Table 2. Our model achieves the best METEOR score among all the models. As BLEU only calculate the exact overlapping between generated sentences and reference sentences, lower BLEU score does not necessarily indicate bad quality. In addition, every test case only has one reference text, which makes automatic evaluation more difficult to measure the real quality of multiple generated sentences. Since METEOR compares the generated sentences with the reference text using not only exact word matching but also stemming and synonym matching, we believe it measures the generation quality more accurately.

For human evaluation, we randomly select 100 input sentences from the test set of each dataset and get the generated results of different models for these inputs. We follow the human evaluation guideline in (Li et al., 2018). The sentence pairs are scored for two aspects of the generated results: *fluency* (whether the generated sentence is fluent and grammatical) and *consistency* (the meaning similarity between the input sentence and the generated sentence). Each output is given two ratings, scaling from 1 to 5, on fluency and consistency separately. An input and outputs generated for the input from different systems form a test group. The outputs from different systems in the test group are shuffled. The test groups are randomly assigned to annotators. Every test group is evaluated by two annotators and scores for each output are averaged. The agreement between annotators is moderate (kappa=0.54). The final score for each system is the average score for all outputs. Table 4 shows our model achieves better scores in both meaning similarity and fluency than baseline models.

Source	How do you know you 're in love again ?
PG-BS	How do you know you are in love with someone ? How do you know if you 're in love with someone ? How do you know if you are in love with someone ?
ours	How do you know if you 're in love with someone ? What are some ways to know that you are in love with someone ? How can you know if you 're in love ?
Source	How does this demonetisation of 500 & 1000 rupee notes help to reduce the price of real estate ?
PG-BS	What will be the effect of banning 500 and 1000 notes on the Indian economy ? What will be the effect of banning 500 and 1000 notes on the stock markets ? What will be the effect of banning 500 and 1000 notes on the real estate sector ?
ours	What will be the effect of banning 500 and 1000 notes on real estate sector in India ? How does the declaration that Rs 500 and Rs 1000 notes would not be accepted as valid transactions affect real estate in India ? How can the ban of 500 and 1000 rupee notes help in curbing the price of real estate ?
Source	I want to write a book - where should I start ?
PG-BS	I want to write a book , where should I start ? I want to write a book , how should I start ? I want to write a book , where can I start ?
ours	How do I write a book ? What should I do to write a book - where should I start ? I want to write a book , where should I start ?

Table 5: Example paraphrases on Quora

In terms of diversity, our approach outperforms other methods by a large margin. VAE-SVG-eq, GAN and D-PAGE do not show better results than beam search. As discussed in (Xu et al., 2018b), beam search guarantees that multiple generated sentences are different from each other by at least one word, which yields better diversity. GAN methods often suffer from mode collapse problems and generate similar outputs. For VAE-SVG-eq, different random noise sampled from a Gaussian prior is used identically in training, so different noise inputs may produce the same sentences. The generator with IRL and DP-GAN have no obvious improvement in diversity in our task. Although the generator with IRL and DP-GAN also claim to improve diversity, their goal is essentially different from ours. The generator with IRL was proposed to tackle the unconditional text generation problem, where texts are generated without any input or condition. DP-GAN generates a more informative output for each input to avoid repetitive response for different inputs. In the experiment, we notice that VAE-SVG-eq and GAN generate similar or even identical sentences according to different random noise, which is consistent with the results posted in (Xu et al., 2018b). D-PAGE may able to find different patterns for very short sentences or phrases, but results in our experiments show that D-PAGE does not separate different patterns in longer sentences and the generated sentences lack diversity.

Table 3 shows scores on Twitter dataset. Our model achieves the best BLEU score and outperforms all baselines for METEOR. The human evaluation results presented in table 4 also show that our model produces better quality paraphrases. For diversity, our model again performs considerably better than other models. Since part of sentence pairs in Twitter dataset may be mislabelled by the labeling algorithm, it is harder to generate high-quality paraphrases on Twitter. Paraphrases generated by our model have not only more diverse expression but also better quality compared to those generated by other models.

Table 5 demonstrates some examples generated by pointer-generator using beam search and our model. Although pointer-generator indeed generates different sentences, it only yields little diversity. Sentences produced by pointer-generator can be very similar, many of them only differ from others by one word. Compared to sentences generated by pointer-generator, those produced by our model has obvious differences between each other. In the first example, our model uses different types of interrogative. In contrast, pointer-generator generates sentences that are almost the same. In the second example, pointer-generator generates similar sentences and yields obvious meaning changes from the input sentence. The sentences generated by our model are in different expressions and more relevant to the input sentence. In the third example, the sentences generated by pointer-generator show minor variations, while those generated by our model present richer expressions.

6 Conclusion

In this paper, we present a novel approach to generate diverse paraphrases. Multiple generators are pretrained in the same way, then learn different generation pattern and enhance generation quality via reinforcement learning. Paraphrase discriminator evaluates the meaning similarity between two sentences, generator discriminator distinguishes sentences generated from different generators. Experiments show that our model has better results concerning both diversity and quality. In the future, we would like to apply our framework to increase reference texts for automatic evaluation or augment training data for text classification.

Acknowledgments

The work is sponsored by Huawei Innovation Research Program. The corresponding authors Weinan Zhang and Yong Yu are also supported by NSFC (61702327, 61772333, 61632017). We thank the anonymous reviewers for their valuable comments.

References

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

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734.

Bo Dai, Sanja Fidler, Raquel Urtasun, and Dahua Lin. 2017. Towards diverse and natural image descriptions via a conditional gan. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2970–2979.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12(Jul):2121–2159.

Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2014. Open question answering over curated and extracted knowledge bases. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 1156–1165. ACM.

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*.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 410–413. Association for Computational Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural computation*, 9(8):1735–1780.

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)*, pages 1875–1885.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234.

Alon Lavie and Abhaya Agarwal. 2007. 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*, pages 228–231. Association for Computational Linguistics.

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562*.

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

Nitin Madnani and Bonnie J Dorr. 2010. Generating phrasal and sentential paraphrases: A survey of data-driven methods. *Computational Linguistics*, 36(3):341–387.

Kathleen R McKeown. 1983. Paraphrasing questions using given and new information. *Computational Linguistics*, 9(1):1–10.

Ramesh Nallapati, Bowen Zhou, Cicero dos Santos, Ça glar Gulçehre, and Bing Xiang. 2016. Abstractive text summarization using sequence-to-sequence rnns and beyond. *CoNLL 2016*, page 280.

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.

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*.

Chris Quirk, Chris Brockett, and Bill Dolan. 2004. Monolingual machine translation for paraphrase generation.

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

Steven J Rennie, Etienne Marcheret, Youssef Mroueh, Jarret Ross, and Vaibhava Goel. 2017. Self-critical sequence training for image captioning. In *CVPR*, volume 1, page 3.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1073–1083.

Iulian Vlad Serban, Tim Klinger, Gerald Tesauro, Kartik Talamadupula, Bowen Zhou, Yoshua Bengio, and Aaron C Courville. 2017. Multiresolution recurrent neural networks: An application to dialogue response generation. In *AAAI*, pages 3288–3294.

Zhan Shi, Xinchi Chen, Xipeng Qiu, and Xuanjing Huang. 2018. Towards diverse text generation with inverse reinforcement learning. *arXiv preprint arXiv:1804.11258*.

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

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 203–207. Association for Computational Linguistics.

Jingjing Xu, Xuancheng Ren, Junyang Lin, and Xu Sun. 2018a. Dp-gan: diversity-promoting generative adversarial network for generating informative and diversified text. *arXiv preprint arXiv:1802.01345*.

Qiongkai Xu, Juyan Zhang, Lizhen Qu, Lexing Xie, and Richard Nock. 2018b. D-page: Diverse paraphrase generation. *arXiv preprint arXiv:1808.04364*.

Pengcheng Yin, Nan Duan, Ben Kao, Junwei Bao, and Ming Zhou. 2015. Answering questions with complex semantic constraints on open knowledge bases. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, pages 1301–1310. ACM.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve smt-based paraphrasing model. *Proceedings of ACL-08: HLT*, pages 1021–1029.

Yaoming Zhu, Sidi Lu, Lei Zheng, Jiaxian Guo, Weinan Zhang, Jun Wang, and Yong Yu. 2018. Texygen: A benchmarking platform for text generation models. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, pages 1097–1100. ACM.