Paraphrase Generation with Deep Reinforcement Learning

Zichao Li[1], Xin Jiang[1], Lifeng Shang[1], Hang Li[2]
[1]Noah's Ark Lab, Huawei Technologies
{li.zichao, jiang.xin, shang.lifeng}@huawei.com
[2]Toutiao AI Lab
lihang.lh@bytedance.com

Abstract

Automatic generation of paraphrases for a given sentence is an important yet challenging task in natural language processing (NLP), and plays a key role in a number of applications such as question answering, information retrieval and dialogue. In this paper we present a deep reinforcement learning approach to paraphrase generation. Specifically, we propose a new model for the task, which consists of a *generator* and a *teacher*. The generator, built on the sequence-to-sequence learning framework, can generate paraphrases given a sentence. The teacher, modeled as a deep neural network, can decide whether the sentences are paraphrases of each other. After construction of the generator and teacher, the generator is further fine-tuned by reinforcement learning in which the reward is given by a teacher. Empirical study shows that the teacher can provide precise supervision to the generator, and guide the generator to produce more accurate paraphrases. Experimental results demonstrate the proposed model outperforms the state-of-the-art methods in paraphrase generation in both automatic evaluation and human evaluation.

1 Introduction

Paraphrases refer to texts that convey the same meaning with different expressions. For example, "*how far is Earth from Sun*", "*what is the distance between Sun and Earth*" and "*how many miles is it from Earth to Sun*" are paraphrases. Paraphrase generation is an important task in NLP, which can be a key component in many scenarios such as retrieval-based question answering, query reformulation for web search, data augmentation for task-oriented dialogue. However, due to the flexibility and diversity of natural language, automatically generating accurate and diverse paraphrases for a given sentence is still very challenging. Traditional symbolic approaches to paraphrase generation include rule-based methods (McKeown, 1983), thesaurus-based methods (Bolshakov & Gelbukh, 2004; Kauchak & Barzilay, 2006) and statistical machine translation (SMT) based methods (Quirk et al., 2004; Zhao et al., 2008, 2009).

Recently, neural network based sequence-to-sequence (Seq2Seq) learning has made remarkable success in various NLP tasks, including machine translation (Cho et al., 2014; Sutskever et al., 2014; Bahdanau et al., 2015; Wu et al., 2016), short-text conversation (Shang et al., 2015; Vinyals & Le, 2015), text summarization (Rush et al., 2015; Chopra et al., 2016; Nallapati et al., 2016), question answering (Yin et al., 2015; He et al., 2017). Moreover, equipped with the recently developed techniques such as attention mechanism (Bahdanau et al., 2015), copying mechanism (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017) and coverage model (Tu et al., 2016), Seq2Seq models become even more powerful to fulfill the specific requirements of individual applications. Paraphrase generation can naturally be formulated as a sequence-to-sequence learning problem, given a collection of pairs of

paraphrases as training data. There are several existing works (Cao et al., 2017; Prakash et al., 2016; Gupta et al., 2017) falling into this category, which have demonstrated the superior performance of the neural approach to paraphrase generation.

A sequence-to-sequence model is usually trained by maximizing the log-likelihood of predicting the output sequences given the input sequences, which can induce discrepancy between training and testing. To tackle this problem, Ranzato et al. (2015) propose using reinforcement learning (RL), specifically the REINFORCE algorithm (Williams, 1992), to directly optimize the sequence-level metric such as BLEU (Papineni et al., 2002) or ROUGE (Lin, 2004). However, using these metrics as reward functions for paraphrasing can be problematic, because all the metrics are calculated based on the lexical similarity to the human references, which may not perfectly represent paraphrase relations. It is likely that a correctly generated sequence will get a low reward due to lexical mismatch. As a result, the trained model can be sub-optimal.

In this work, we also employ reinforcement learning to guide the training of the generator for paraphrase generation through policy gradient. Instead of using an evaluation metric, we exploit a deep matching network, which can discriminate positive and negative samples as paraphrases, to model the reward function. To be specific, we adopt the attentive Seq2Seq architecture with copy and coverage mechanisms (Vinyals et al., 2015; Gu et al., 2016; See et al., 2017) as the generator, and the deep matching network based on the decomposable attention model (Parikh et al., 2016) as the teacher. We argue that a well-trained matching network is more suitable for providing accurate training signals, and thus can reduce the training variances and further improve the accuracy of the generator. Besides, the matching network is trained with both positive and negative examples, which means that the generator can be fine-tuned with more data. We evaluate our approach on the Quora question pair dataset through both automatic and human assessments, and report the evaluation results. The results indicate that our model can achieve the best performances among the existing methods. It should be noted that the proposed approach is not limited to paraphrase generation and can be readily applied into other sequence-to-sequence tasks such as machine translation and short text conversation.

2 Background

Paraphrase generation can be regarded as a sequence-to-sequence transformation problem. Given an input sequence of words $X = [x_1, \ldots, x_L]$ with length L, we consider generating another output sequence of words $Y = [y_1, \ldots, y_T]$ with length T that has the same meaning as X. There are different levels of granularities from phrase to passage, and we address the sentence-level paraphrasing in this work. We denote a pair of paraphrases as (X, Y). For simplicity we use $Y_{1:t}$ to denote the subsequence ranging from 1 to t, and use $\hat{Y}$ to denote the model-predicted sequence.

2.1 Neural Sequence-to-Sequence Models with Attention

In this work, the task of paraphrase generation is defined as a sequence-to-sequence learning problem which aims to learn the mapping from one sequence to another. Specifically, the Seq2Seq model is built based on the encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014), both of which are parameterized by Recurrent Neural Networks (RNN). After reading the input sequence X, the encoder RNN transforms it to a sequence of hidden states $H = [h_1, \ldots, h_L]$, where $h_i = f_h(x_i, h_{i-1})$. Conditioned on the states and previous word y_{t-1}, the decoder RNN predicts the next word by sampling $\hat{y}_t \sim p(y_t|Y_{1:t-1}, X) = g(s_t, c_t, y_{t-1})$, where $s_t = f_s(s_{t-1}, y_{t-1}, c_t)$ is the decoder state and c_t is the *context vector*. Attention mechanism (Bahdanau et al., 2015) is introduced to

compute the context vector as the weighted sum of encoder states:

$$c_t = \sum_{i=1}^{L} \alpha_{ti} h_i, \;\; \alpha_{ti} = \frac{\exp \eta(s_{t-1}, h_i)}{\sum_{k=1}^{L} \exp \eta(s_{t-1}, h_k)} \tag{1}$$

where α_{ti} represents the attention weight and η is the *attention function*, which can be a feed-forward neural network and jointly trained within the whole network.

Given training data (X, Y), Seq2Seq model is trained by maximizing the log-likelihood:

$$\mathcal{L}_{\text{seq2seq}}(\theta) = \sum_{t=1}^{T} \log p_\theta(y_t | Y_{1:t-1}, X). \tag{2}$$

Note that when computing conditional probability $p_\theta(y_t|Y_{1:t-1}, X)$, one conventionally chooses the previous word y_{t-1} in the ground-truth rather than the model generated word $\hat{y}_{t-1}$, in order to avoid compounding errors along the sequence during training. This method is called *teacher forcing*.

2.2 Incorporating Copying and Coverage Mechanism

Paraphrasing often requires copying some words from the input sequence, for instance named entities. The ability of copying has been developed for sequence-to-sequence learning in previous work (Vinyals et al., 2015; Gu et al., 2016; Cheng & Lapata, 2016; See et al., 2017). Specifically, in addition to sampling words from a fixed-size vocabulary, the decoder can also select words directly from the input sequence. The decoding probability for the next word is given by a mixture model:

$$p_\theta(y_t|Y_{1:t-1}, X) = q(s_t, c_t, y_{t-1}) g(s_t, c_t, y_{t-1}) + (1 - q(s_t, c_t, y_{t-1})) \sum_{i:y_t=x_i} \alpha_{ti}, \tag{3}$$

where $q(s_t, c_t, y_{t-1})$ is a binary classifier governing the probability switching between the generation mode and the copy mode.

Besides, we find in our preliminary experiments that the paraphrase generator can suffer from the problem of repeating words. Therefore, we add the coverage loss (Tu et al., 2016) to the objective function to deal with the problem:

$$\mathcal{L}_{\text{copy+coverage}}(\theta) = \sum_{t=1}^{T} [\log p_\theta(y_t|Y_{1:t-1}, X) - \lambda_c \sum_{i=1}^{L} \min(\alpha_{ti}, \sum_{\tau=1}^{t-1} \alpha_{\tau i})]. \tag{4}$$

2.3 Neural Networks for Matching Sentences

Another problem regarding to paraphrasing is paraphrase identification, which is to decide whether a pair of sentences are paraphrases. The problem naturally boils down to determining the matching degree between two sentences. A variety of learning techniques have been developed for matching short texts, from linear models (e.g., Wu et al. (2013)) to neural network based models (e.g., Socher et al. (2011); Hu et al. (2014)). Our idea in this work is to first train a matching model that can identify paraphrases, and then tune the paraphrase generator based on the feedback from the matching model. In our implementation we choose a simple yet effect neural architecture, called *decomposable-attention* model (Parikh et al., 2016), to compute the similarity between two sentences. Given two input sentences X and Y, the model contains three steps as follow.

Attend Two sentences are inter-attended by each other:

$$\begin{aligned}\bar{x}_i &= \sum_{j=1}^{T} \alpha_{ji} e(y_j), \;\; \alpha_{ji} = \frac{\exp e(x_i)^T e(y_j)}{\sum_{k=1}^{T} \exp e(x_i)^T e(y_k)}, \\ \bar{y}_j &= \sum_{i=1}^{L} \beta_{ji} e(x_i), \;\; \beta_{ji} = \frac{\exp e(x_i)^T e(y_j)}{\sum_{k=1}^{L} \exp e(x_k)^T e(y_j)}.\end{aligned} \tag{5}$$

Here $e(\cdot)$ is the word embedding vector of dimension d, and $\bar{x}_i$ is the inter-attentive vector for x_i.
Compare For each word, the embedding vector is merged with its attentive vector, through a feed-forward network f_c:

$$\mathbf{x}_i = f_c(e(x_i), \bar{x}_i), \;\; \mathbf{y}_j = f_c(e(y_j), \bar{y}_j) \tag{6}$$

Aggregate The vectors in the two sequences are first aggregated by summation and then followed by a linear classifier f_a:

$$z = f_a\left(\sum\nolimits_{i=1}^{L} \mathbf{x}_i, \sum\nolimits_{j=1}^{T} \mathbf{y}_j\right), \tag{7}$$

where z is the matching degree between X and Y. We refer the readers to the original paper for more details.

In addition, we add fixed *positional encodings* to the word embedding vectors as proposed in Vaswani et al. (2017) , so as to incorporate the order information of the sentence:

$$e'_k(i) = \begin{cases} \sin(i/10000^{k/d}) \text{ , if } k \equiv 0 \text{ mod } (2), \\ \cos(i/10000^{(k-1)/d}) \text{ , if } k \equiv 1 \text{ mod } (2), \end{cases} \tag{8}$$

where $e'(i)$ stands for the position encoding vector at position i, and $e'_k(i)$ denote its value on the k-th dimension.

3 Models

Although demonstrating fairly good results on paraphrasing (e.g., Cao et al. (2017); Prakash et al. (2016)), conventional sequence-to-sequence learning still has several limitations: 1) it usually requires a large parallel corpus to train a model, which may not be available in practice; 2) learning with teacher forcing, the discrepancy between training and prediction (also referred as to *exposure bias*) can yield errors quickly accumulating along the generated sequence (Bengio et al., 2015; Ranzato et al., 2015); 3) using log-likelihood as the objective is not optimal for the problem. The ultimate goal of paraphrasing is to generate multiple sentences that have the same meaning as the input sentence, but not to perform the most plausible sequence-to-sequence translation. To attack these problems, we propose further optimizing the Seq2Seq model under the framework of reinforcement learning (RL). Instead of using a lexicon-based evaluation metric as the objective, we additionally learn a neural matching network, referred to as *teacher*, that can provide more reliable signal to fine-tune the *generator*. We denote the generator as G_θ and the teacher as M_ϕ, where θ and ϕ represent their parameters respectively. Figure 1 gives on overview of our approach.

3.1 Reinforcement Learning for Sequence Prediction

As mentioned in 2.1, the generator G_θ is trained to maximize the conditional likelihood of the output sequence given the input sequence. This makes the learnt generator suboptimal. The problem

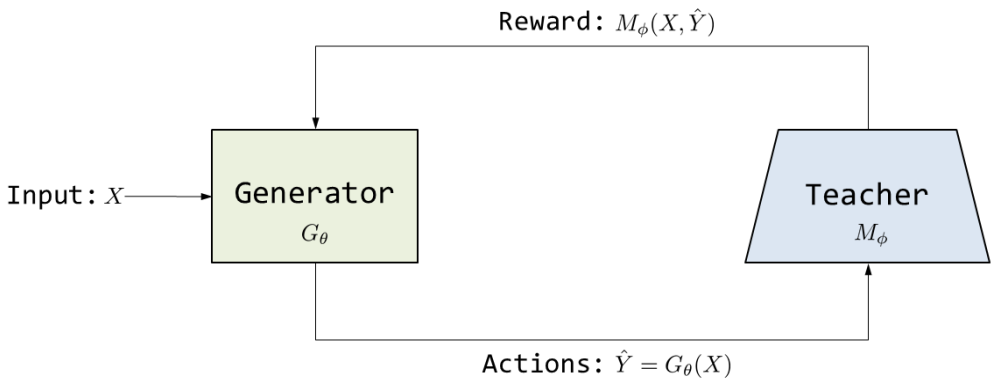

Figure 1: Framework of Deep Reinforcement Learning for Paraphrase Generation.

is tackled by addressing the sequence generation problem with the reinforcement learning framework (Ranzato et al., 2015; Bahdanau et al., 2016). In such framework, the generator G_θ is trained to maximize the future expected reward, which can be obtained in various manner. One common choice for reward function is an evaluation metric such as BLEU or ROUGE, calculated by comparing the generated sequence against the reference. This paper is along the line of research, and we explain the training method in this section.

In the RL setting, generation of the next word is defined as an *action*, and the decoding probability $p_\theta(y_t|Y_{1:t-1}, X)$ induces a stochastic *policy*. Let r_t denote the *reward* at time step t and $R = \sum_{t=1}^{T} r_t$ denote the cumulative reward (the *return*), and the goal of RL is to find a policy (accordingly the generator) that maximizes the expected return:

$$\mathcal{L}_{RL}(\theta) = \mathbb{E}_{p_\theta(Y_{1:T}|X)} \sum_{t=1}^{t=T} r_t(y_t; X, Y_{1:t-1}) \tag{9}$$

In the task of sequence generation, the reward is usually received at the end of sequence, meaning that $r_t = 0, t < T$ and $R = r_T$. This will lead to sparse reward signals and make training less efficient. Inspired by the idea of reward shaping (Ng et al., 1999; Bahdanau et al., 2016), we use the value function as a surrogate, which is defined as the expected cumulative reward at each time step $Q(y_t; X, \hat{Y}_{1:t-1}) = \mathbb{E}_{p_\theta(Y_{t+1:T}|\hat{Y}_{1:t}, y_t, X)} R(Y)$. The value function can be estimated by aggregating the Monte-Carlo simulations at each time step:

$$Q(y_t; X, \hat{Y}_{1:t-1}) \approx \begin{cases} \frac{1}{N} \sum_{n=1}^{N} R([\hat{Y}_{1:t-1}, y_t, \widehat{Y}^n_{t+1:T}]), & t < T \\ R([\hat{Y}_{1:t-1}, y_t]), & t = T. \end{cases} \tag{10}$$

Here $\{\widehat{Y}^1_{t+1:T}, ...\widehat{Y}^N_{t+1:T}\} \sim p_\theta(Y_{t+1:T}|\hat{Y}_{1:t-1}, y_t, X)$ denotes a set of simulated sequences, which are randomly sampled starting from time step $t+1$ conditioned on the current states and action.

In our experiment, we try to use ROUGE-2 score as the reward, namely $R(\hat{Y}) = \text{ROUGE-2}(\hat{Y}, Y)$, where Y denotes the ground-truth reference.

According to the policy gradient theorem (Williams, 1992; Sutton et al., 2000), the gradients of the expected return can be estimated by

$$\nabla_\theta \widehat{\mathcal{L}}_{RL}(\theta) = \sum_{t=1}^{T} \nabla_\theta \log p_\theta(\hat{y}_t|\hat{Y}_{1:t-1}, X) Q(\hat{y}_t; X, \hat{Y}_{1:t-1}) \tag{11}$$

In order to reduce the variance of the above gradient estimator, a baseline function b is subtracted from the value function (Williams, 1992; Ranzato et al., 2015) as follows:

$$\nabla_\theta \widehat{\mathcal{L}}_{RL}(\theta) = \sum_{t=1}^{T} \nabla_\theta \log p_\theta(\hat{y}_t|\hat{Y}_{1:t-1}, X)(Q(\hat{y}_t; X, \hat{Y}_{1:t-1}) - b) \tag{12}$$

In our implementation, instead of using another separate neural network to estimate the baseline function, we simply maintain a moving average, that is

$$b_m = \gamma \overline{Q}_{m-1} + (1-\gamma) b_{m-1},\ b_1 = 0 \tag{13}$$

where $\overline{Q}_{m-1}$ denotes the average reward within one batch at the $(m-1)$-th iteration during training.

3.2 Learning Reward with Deep Matching Network

Reward function is crucial to the effectiveness of reinforcement learning for sequence prediction. However, previously used reward functions such as BLEU or ROUGE are lexicon-based similarities, which may lead to unfair judgement due to word mismatch. For instance, an input sentence "*how far is Earth from Sun*" can either be paraphrased as "*what is the distance between Sun and Earth*" or "*how many miles is it from Earth to Sun*", but the latter will get very low ROUGE score (thus negative reward) given the former as the reference. This inevitably increases the variance of the gradient estimator (11) and makes the training less effective. Moreover, since the above metrics are calculated based on reference sentences, the RL training can only be performed on the parallel data (as in supervised learning), which can be expensive to obtain in practical applications.

Therefore, in this paraphrasing task, we design a neural network based matching model as the reward function, i.e. $R(\hat{Y}) = M_\phi(X, \hat{Y})$, which is learnt to measure the semantic similarity between sentences. In particular, when a pair of the input sentence X and the generated sentence $\hat{Y}$ is fed, the matching network $M_\phi(X, \hat{Y})$ can output how likely this pair being paraphrases. As a teacher of the generator, M_ϕ is expected to be more accurate and smooth than the lexicon-based measures. After the matching network is well learnt, the reward to the generator can be obtained without need of the ground-truth Y. As a result, we are able to fine-tune the generator by simply using plain (unparallel) text, which can be much more easily obtained.

The implementation of the matching network in this work is based on the *decomposable-attention* architecture (Parikh et al., 2016), as detailed in section 2.3. Given a set of positive and negative paraphrasing pairs, we train M_ϕ with the binary cross-entropy loss:

$$\mathcal{J}_{\text{pointwise}}(\phi) = -\log M_\phi(X, Y) - \log(1 - M_\phi(X, Y^-)), \tag{14}$$

where Y^- represents a sentence that is not a paraphrase of X.

Alternatively we can use the margin-based ranking loss:

$$\mathcal{J}_{\text{pairwise}}(\phi) = \max(0, 1 + M_\phi(X, Y^-) - M_\phi(X, Y)). \tag{15}$$

In our framework, the accuracy of the teacher will considerably affect the performance of the generator. Training the matching network also requires additional labeled data, either pointwise or pairwise. Nevertheless, training the discriminative model is empirically easier than training the generative model in general.

With a well-trained matching network M_ϕ, we can further train the generator G_θ by reinforcement learning with the output from M_ϕ. The objective function and training procedure remain the same as in section 3.1. The only difference is to use a different reward, or accordingly the value function:

$$Q(y_t; X, \hat{Y}_{1:t-1}) = \begin{cases} \frac{1}{N}\sum_{n=1}^{n=N} M_\phi(X, [\hat{Y}_{1:t-1}, y_t, \widehat{Y}^n_{t+1:T}]), & t < T \\ M_\phi(X, [\hat{Y}_{1:t-1}, y_t]), & t = T. \end{cases} \tag{16}$$

Note that the calculation of the value function only requires sequence X as input, and according to equation (11) the gradients can be determined without the ground-truth output Y. This allows the generator being trained with unlabeled text, which can potentially enhance its generalization ability. From another point of view, combined with the matching network, the Seq2Seq model can make full use of information other than a parallel corpus, such as negative examples, preferences relations, and unsupervised data.

We call our method RbM (Reinforced by Matching). In RbM model, the teacher M_ϕ is expected to capture various paraphrasing patterns and to make precise judgements on whether two sentences are paraphrases. Trough maximizing the expected return induced from M_ϕ, the generator G_θ can learn to generate more accurate paraphrases. The detailed training procedure is shown in Algorithm 1.

4 Experiment

4.1 Datasets

We validate our method with the Quora dataset. The Quora dataset is a newly released dataset with more than 400K pairs of questions [1]. Each question pair is labeled whether they are *duplicated* (i.e., they are paraphrases) or not. Around 140K of question pairs are labeled as duplicated, while the other pairs are not. We use the duplicated questions to train the generator G_θ. In particular, we split the questions into subsets for training (100K), testing (30K), and validation (3K), respectively. In the RL phrase of the generator, besides duplicated pairs, non-duplicated pairs are also used as the unparallel data for training. For the matching network M_ϕ, we use both the *duplicated* and *non-duplicated* pairs for training (200K), testing (40K) and validation (4K), respectively. To ensure a fair evaluation, we exclude the testing and validation data of the generator from the training data of matching network.

4.2 Implementation Details

4.2.1 Paraphrase Generator

For paraphrase generator G_θ, we maintain a fixed-size vocabulary of 10K shared by input and output words, and truncate all the sentences longer than 20 words. The word embeddings with 128 dimensions are trained from scratch. We employ the Long-Short-Term-Memory (LSTM) (Hochreiter & Schmidhuber, 1997) unit of 256 dimension for the encoder and the decoder, and use bidirectional RNN for the encoder. Generator is trained on multiple NVIDIA Tesla K80 GPUs using TensorFlow (Abadi et al., 2016), with the batch size of 80. We use Adadgrad optimizer (Duchi et al., 2011) in both stages of supervised pre-training and reinforcement learning. In supervised pre-training, we set the learning rate as 0.1 and initial accumulator as 0.1. The maximum norm of gradient is set as 2. We decrease the learning rate to half during the pre-training of G_θ. We set λ_c in equation (4) to 1. During the RL training, the learning rate is decreased to 0.001 and the size of Monte-Carlo sample is $N = 16$. We

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

Algorithm 1: Training Procedure of Paraphrase Generator Reinforced by Matching Network

Input : A corpus of paraphrase pairs $\{(X, Y)\}$, a corpus of non-paraphrase pairs $\{(X, Y^-)\}$, a corpus of (unparallel) sentences $\{X\}$.

Output: Generator $G_{\theta'}$

Train the matching network M_ϕ with $\{(X, Y)\}$ and $\{(X, Y^-)\}$;

Pre-train the generator G_θ with $\{(X, Y)\}$;

Init $G_{\theta'} := G_\theta$;

while *not converge* **do**

  Sample a sentence $X = [x_1, \ldots, x_L]$ from the unparallel corpus;

  Generate a sentence $\hat{Y} = [\hat{y}_1, \ldots, \hat{y}_T]$ according to $G_{\theta'}$ given input X;

  Set the gradient $g_{\theta'} = 0$;

  for $t = 1$ **to** T **do**

    Run N Monte Carlo simulations according to $G_{\theta'}$:

$$\{\widehat{Y}^1_{t+1:T}, ...\widehat{Y}^N_{t+1:T}\} \sim p_{\theta'}(Y_{t+1:T}|\hat{Y}_{1:t-1}, \hat{y}_t, X)$$

    Compute the value function:

$$Q(\hat{y}_t; X, \hat{Y}_{1:t-1}) = \frac{1}{N} \sum_{n=1}^{n=N} M_\phi(X, [\hat{Y}_{1:t-1}, \hat{y}_t, \widehat{Y}^n_{t+1:T}])$$

    Accumulate the gradient:

$$g_{\theta'} := g_{\theta'} + \nabla_\theta \log p_{\theta'}(\hat{y}_t|\hat{Y}_{1:t-1}, X)(Q(\hat{y}_t; X, \hat{Y}_{1:t-1}) - b)$$

  end

  Update $G_{\theta'}$ using the gradient $g_{\theta'}$ with learning rate γ:

$$G_{\theta'} := G_{\theta'} + \gamma g_{\theta'}$$

end

Return $G_{\theta'}$

set γ in equation (13) to 0.1. To make the training more stable, we use the ground-truth with reward of 0.001.

4.2.2 Matching Network

On the Quora dataset, we use binary cross-entropy loss (14) since pointwise labeled data are available. Unlike in training generator, the word embedding for M_ϕ is pre-trained by word2vec and fine-tuned afterwards. Specifically, we use the GoogleNews 300-dimension word vectors [2], and map them to 200 dimensions as Parikh et al. (2016) did. The matching network M_ϕ is trained on multiple NVIDIA Tesla K80 GPUs using TensorFlow, with the batch size of 32. We use ReLU layers with dropout rate of 0.2, and employ Adagrad optimizer (Duchi et al., 2011) with learning rate of 0.05. To make the model order sensitive, we inject the position encodings into the word embedding as proposed in Vaswani et al. (2017). We find that it slightly hurts the accuracy of matching but can benefit the generator eventually.

4.3 Results

To compare with our model, we choose three baseline models: a vanilla (attentive) Seq2Seq model, a Seq2Seq model with copy and coverage mechanism and a Seq2Seq model reinforced by ROUGE-2. We conduct both quantitative and qualitative evaluation on different models. As for the quantitative evaluation, we adopt four automatic evaluation metrics: ROUGE-1, ROUGE-2 (Lin, 2004), BLEU (Papineni et al., 2002) and METEOR (Lavie & Agarwal, 2007). Table 1 and table 2 show the performances of the models under different beam-search strategies. We find that our model (RbM) outperforms all the existing models in terms of all the evaluation measures.

Table 1: Performances of different paraphrase generators (beam size=1) on Quora dataset.

Models	ROUGE-1	ROUGE-2	BLEU	METEOR
Vanilla Seq2Seq	58.50	30.70	35.91	25.16
Seq2Seq with Copy and Coverage	61.40	35.12	39.90	29.63
Seq2Seq Reinforced by ROUGE-2 (RbR)	62.97	36.6	41.39	29.64
Seq2Seq Reinforced by Matching (RbM)	**63.71**	**37.55**	**42.33**	**31.54**

Table 2: Performances of different paraphrase generators (beam size=10) on Quora dataset.

Models	ROUGE-1	ROUGE-2	BLEU	METEOR
Vanilla Seq2Seq	58.77	31.47	36.55	26.28
Seq2Seq with Copy and Coverage	61.96	36.07	40.55	30.21
Seq2Seq Reinforced by ROUGE-2 (RbR)	63.35	37.33	41.83	30.96
Seq2Seq Reinforced by Matching (RbM)	**63.81**	**37.81**	**42.49**	**31.87**

In addition to the automatic evaluation, we also conduct human evaluation. We randomly select 300 source sentences from the test data and generate paraphrases using different models. The pairs of paraphrases are then blindly partitioned to seven human assessors, who are asked to rate each sentences pair according to the following two criteria: *relevance* (the paraphrase sentence is semantically close to

[2] https://code.google.com/archive/p/word2vec/

Figure 2: Examples of the generated paraphrases by RbM model on Quora dataset.

Questions	Top Generated Paraphrases
why do people ask stupid questions on quora that could be easily answered by google ?	why do so many people ask google - able questions on quora ?
	why do people ask quora questions which can be answered easily by google ?
	why do people ask questions on quora that are easily answerable via a quick internet search ?
is there any chance that donald trump will win this election ?	what are the chances that donald trump will be our next president ?
	what are the chances that donald trump will win the election ?
	what are the chances of donald trump winning the 2016 presidential election ?
how will abolishing rs . 500 and rs . 1000 notes reduce corruption and identifying black money ?	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 ?
	how can black money brought out by discontinuing 500 and 1000 notes ?

the source sentence) and *fluency* (the paraphrase sentence is fluent as a natural language sentence, and the grammar is correct). Hence each assessor gives two scores to each paraphrase, both ranking from 1 to 5. To reduce the evaluation variance, each pair is rated by two assessors, and then averaged as the final judgement. Table 3 demonstrates the average ratings for each model, including the ground-truth references. Our model gets the better scores in terms of both relevance and fluency over the baseline models, and their differences on relevance scores are statistically significant (paired t-test, p-value < 0.01). Figure 2 gives some examples of top generated paraphrases by the RbM model on the test dataset.

Table 3: Human evaluation results on Quora dataset.

Models	Relevance	Fluency
Seq2Seq with Copy and Coverage	3.23	4.55
Seq2Seq Reinforced by ROUGE-2 (RbR)	3.56	4.61
Seq2Seq Reinforced by Matching (RbM)	**4.00**	**4.62**
Ground-truth Reference	4.69	4.95

5 Related Work

5.1 Sequence-to-Sequence Learning

The encoder-decoder framework for sequence-to-sequence learning is first proposed by Sutskever et al. (2014); Cho et al. (2014) and it can map one sequence to another with variable length. It has been widely applied in natural language generation (NLG) tasks, such as machine translation (e.g., Wu et al. (2016)), dialogue generation (e.g., Shang et al. (2015); Vinyals & Le (2015)), document summarization (e.g., Rush et al. (2015)), and question answering (e.g., (Yin et al., 2015)). In the vanilla Seq2Seq model, the source sequence is compressed to a fixed-length vector, no matter how long the sequence is. Thus, it suffers from information loss, especially when the source sequence is long. Bahdanau et al. (2015) tackles this problem by attention mechanism. At each decoding step,

the model attends to a specific part of the source sequence to capture significant information. Another problem in Seq2Seq model is existence of out-of-vocabulary (OOV) words. In most of the NLG tasks, a fixed-size vocabulary is maintained. Only the most frequent words in the training set is included in vocabulary. Others are considered as OOV and replaced by the UNK token. As a result, the performance of Seq2Seq model is seriously affected. Gu et al. (2016) incorporate copy mechanism to capture the occurrence of rare words like entity names in both source and target sequences. Then other variants of such mechanism for different monolingual NLG tasks are proposed. For instance, See et al. (2017) utilizes copy mechanism in document summarization. In our work, the Seq2Seq model with attention and copy mechanism is employed in the generator. Differing from all the works mentioned above, the generator in our work is trained via reinforcement learning. There are also works formalizing NLG as a reinforcement learning problem, that is, optimizing the generator with respect to an automatic evaluation metric directly (Ranzato et al., 2015; Bahdanau et al., 2016). While in this work, we employ an auxiliary deep matching model instead of the evaluation metrics, since it can capture the semantic similarity better.

5.2 Deep Matching Model for Semantic Matching

Many deep matching models are proposed for semantic matching. Most of them can also be adapted to paraphrase identification, including but not limited to RNN-based methods (Wang & Jiang, 2015), CNN-based methods (Hu et al., 2014) and attention-based methods (Parikh et al., 2016). We use the attention-based matching model (Parikh et al., 2016) in this work, since it is more efficient and capable of capturing the phrase-level similarity. Inspired by Vaswani et al. (2017), we also incorporate word order information by positional encoding.

5.3 Generative Adversarial Nets for NLP

Recently a new framework to train an unsupervised generative model called Generative Adversarial Net (GAN) (Goodfellow et al., 2014) is gaining popularity. It contains a discriminator and a generator. The generator in GAN tries to map the distribution of noise to the true distribution of data in the real world, while the discriminator tries to tell whether a sample is generated from the generator or the true distribution. There are applications of such framework to NLP, such as text generation (Yu et al., 2017) and dialogue generation (Li et al., 2017). In this work, the largest difference is that the teacher (corresponding to the discriminator) is kept fixed after pre-training. In GAN, however, the discriminator is trained together with the generator. The teacher (matching model) can make the generator more reliably trained.

5.4 Paraphrase Generation

Besides the traditional symbolic-based approach, there are several works on paraphrase generation with sequence-to-sequence learning. Prakash et al. (2016) employ a residual net in the Seq2Seq model to enlarge the model capacity. Cao et al. (2017) utilize an additional vocabulary to restrict the word candidates during decoding. Gupta et al. (2017) use variational auto-encoder framework to generate more diverse paraphrases. Furthermore, all of these existing works train the generator by teacher forcing. Similar to these works, we pre-train our paraphrase generator in the Seq2Seq framework. However, we formulate paraphrase generation as a reinforcement learning problem. We keep both the generator and teacher extendable, since what we propose is a general framework. Our method does not rely on any auxiliary vocabulary, which is often difficult to obtain in practice. We do not

employ the variational autoencoder (VAE) framework either. Therefore, our model is easier to train, and does not suffer from the problem of KL cost annealing (Bowman et al., 2015). To the best of our knowledge, this work is the first one to employ the generator-teacher framework for sequence-to-sequence learning. This framework can generate more accurate paraphrases and allow us to fully utilize the unlabeled data, which the existing models cannot.

6 Conclusion and Future Work

In this paper we have proposed a deep reinforcement learning approach to paraphrase generation, with the generator and reward function modeled by deep neural networks. With a well-trained matching network to define the reward function, the generator, which is a sequence-to-sequence model, can be fine-tuned to produce more accurate paraphrases. The experiment results demonstrate that the proposed approach can indeed improve the quality of generated paraphrases over the baseline approaches. In the future, we plan to apply this approach into other tasks, such as machine translation and short text conversation.

7 Acknowledgments

This work is supported by China National 973 Program 2014CB340301.

References

Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *OSDI*, volume 16, pp. 265–283, 2016.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *ICLR*, 2015.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. An actor-critic algorithm for sequence prediction. *arXiv preprint arXiv:1607.07086*, 2016.

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

Igor Bolshakov and Alexander Gelbukh. Synonymous paraphrasing using wordnet and internet. *Natural Language Processing and Information Systems*, pp. 189–200, 2004.

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349*, 2015.

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. Joint copying and restricted generation for paraphrase. In *AAAI*, pp. 3152–3158, 2017.

Jianpeng Cheng and Mirella Lapata. Neural summarization by extracting sentences and words. *arXiv preprint arXiv:1603.07252*, 2016.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *EMNLP*, 2014.

Sumit Chopra, Michael Auli, and Alexander M Rush. Abstractive sentence summarization with attentive recurrent neural networks. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 93–98, 2016.

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

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pp. 2672–2680, 2014.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*, 2016.

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. *arXiv preprint arXiv:1709.05074*, 2017.

Shizhu He, Cao Liu, Kang Liu, and Jun Zhao. Generating natural answers by incorporating copying and retrieving mechanisms in sequence-to-sequence learning. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pp. 199–208, 2017.

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

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pp. 2042–2050, 2014.

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

Alon Lavie and Abhaya Agarwal. 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. Association for Computational Linguistics, 2007.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out: Proceedings of the ACL-04 workshop*, volume 8. Barcelona, Spain, 2004.

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

Ramesh Nallapati, Bowen Zhou, Caglar Gulcehre, Bing Xiang, et al. Abstractive text summarization using sequence-to-sequence rnns and beyond. *arXiv preprint arXiv:1602.06023*, 2016.

Andrew Y Ng, Daishi Harada, and Stuart Russell. Policy invariance under reward transformations: Theory and application to reward shaping. In *ICML*, volume 99, pp. 278–287, 1999.

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

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. A decomposable attention model for natural language inference. *arXiv preprint arXiv:1606.01933*, 2016.

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

Chris Quirk, Chris Brockett, and William Dolan. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 conference on empirical methods in natural language processing*, 2004.

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

Alexander M Rush, Sumit Chopra, and Jason Weston. A neural attention model for abstractive sentence summarization. *arXiv preprint arXiv:1509.00685*, 2015.

Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017.

Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. In *Association for Computational Linguistics (ACL)*, pp. 1577–1586, 2015.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Advances in Neural Information Processing Systems*, pp. 801–809, 2011.

Ilya Sutskever, Oriol Vinyals, and Quoc VV Le. Sequence to sequence learning with neural networks. In *NIPS*, pp. 3104–3112, 2014.

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. Policy gradient methods for reinforcement learning with function approximation. In *Advances in neural information processing systems*, pp. 1057–1063, 2000.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. Modeling coverage for neural machine translation. *arXiv preprint arXiv:1601.04811*, 2016.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

Oriol Vinyals and Quoc Le. A neural conversational model. *arXiv preprint arXiv:1506.05869*, 2015.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. Pointer networks. In *Advances in Neural Information Processing Systems*, pp. 2692–2700, 2015.

Shuohang Wang and Jing Jiang. Learning natural language inference with lstm. *arXiv preprint arXiv:1512.08849*, 2015.

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

Wei Wu, Zhengdong Lu, and Hang Li. Learning bilinear model for matching queries and documents. *The Journal of Machine Learning Research*, 14(1):2519–2548, 2013.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144*, 2016.

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. Neural generative question answering. *arXiv preprint arXiv:1512.01337*, 2015.

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

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. Combining multiple resources to improve smt-based paraphrasing model. In *ACL*, pp. 1021–1029, 2008.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pp. 834–842. Association for Computational Linguistics, 2009.