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 from a given sentence is an important yet challenging task in natural language processing (NLP). In this paper, we present a deep reinforcement learning approach to paraphrase generation. Specifically, we propose a new framework for the task, which consists of a *generator* and an *evaluator*, both of which are learned from data. The generator, built as a sequence-to-sequence learning model, can produce paraphrases given a sentence. The evaluator, constructed as a deep matching model, can judge whether two sentences are paraphrases of each other. The generator is first trained by deep learning and then further fine-tuned by reinforcement learning in which the reward is given by the evaluator. For the learning of the evaluator, we propose two methods based on *supervised learning* and *inverse reinforcement learning* respectively, depending on the type of available training data. Experimental results on two datasets demonstrate the proposed models (the generators) can produce more accurate paraphrases and outperform 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 but with different expressions. For example, "*how far is Earth from Sun*", "*what is the distance between Sun and Earth*" are paraphrases. Paraphrase generation refers to a task in which given a sentence the system creates paraphrases of it. Paraphrase generation is an important task in NLP, which can be a key technology in many applications such as retrieval based question answering, semantic parsing, query reformulation in web search, data augmentation for dialogue system. However, due to the complexity of natural language, automatically generating accurate and diverse paraphrases is still very challenging. Traditional symbolic approaches to paraphrase generation include rule-based methods (McKeown, 1983), thesaurus-based methods (Bolshakov and Gelbukh, 2004; Kauchak and Barzilay, 2006), grammar-based methods (Narayan et al., 2016), 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, short-text conversation, text summarization, and question answering (e.g., Cho et al. (2014); Wu et al. (2016); Shang et al. (2015); Vinyals and Le (2015); Rush et al. (2015); Yin et al. (2016)). Paraphrase generation can naturally be formulated as a Seq2Seq problem (Cao et al., 2017; Prakash et al., 2016; Gupta et al., 2018; Su and Yan, 2017). The main challenge in paraphrase generation lies in the definition of the evaluation measure. Ideally the measure is able to calculate the semantic similarity between a generated paraphrase and the given sentence. In a straightforward application of Seq2Seq to paraphrase generation one would make use of cross entropy as evaluation measure, which can only be a loose approximation of semantic similarity. To tackle this problem, Ranzato et al. (2016) propose employing reinforcement learning (RL) to guide the training of Seq2Seq and using lexical-based measures such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) as a reward function. However, these lexical measures may not perfectly represent semantic similarity. It is likely that a correctly generated sequence gets a low ROUGE score due to lexical mismatch. For instance, an input sentence "*how far is Earth from Sun*" can be paraphrased as "*what is the distance between Sun and Earth*", but it will

Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, pages 3865–3878
Brussels, Belgium, October 31 - November 4, 2018. ©2018 Association for Computational Linguistics

get a very low ROUGE score given "*how many miles is it from Earth to Sun*" as a reference.

In this work, we propose taking a data-driven approach to train a model that can conduct evaluation in learning for paraphrasing generation. The framework contains two modules, a generator (for paraphrase generation) and an evaluator (for paraphrase evaluation). The generator is a Seq2Seq learning model with attention and copy mechanism (Bahdanau et al., 2015; See et al., 2017), which is first trained with cross entropy loss and then fine-tuned by using policy gradient with supervisions from the evaluator as rewards. The evaluator is a deep matching model, specifically a decomposable attention model (Parikh et al., 2016), which can be trained by supervised learning (SL) when both positive and negative examples are available as training data, or by inverse reinforcement learning (IRL) with outputs from the generator as supervisions when only positive examples are available. In the latter setting, for the training of evaluator using IRL, we develop a novel algorithm based on max-margin IRL principle (Ratliff et al., 2006). Moreover, the generator can be further trained with non-parallel data, which is particularly effective when the amount of parallel data is small.

We evaluate the effectiveness of our approach using two real-world datasets (Quora question pairs and Twitter URL paraphrase corpus) and we conduct both automatic and human assessments. We find that the evaluator trained by our methods can provide accurate supervisions to the generator, and thus further improve the accuracies of the generator. The experimental results indicate that our models can achieve significantly better performances than the existing neural network based 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 generation based single turn dialogue. Our technical contribution in this work is of three-fold:

1. We introduce the generator-evaluator framework for paraphrase generation, or in general, sequence-to-sequence learning.
2. We propose two approaches to train the evaluator, i.e., supervised learning and inverse reinforcement learning.
3. In the above framework, we develop several techniques for learning of the generator and evaluator.

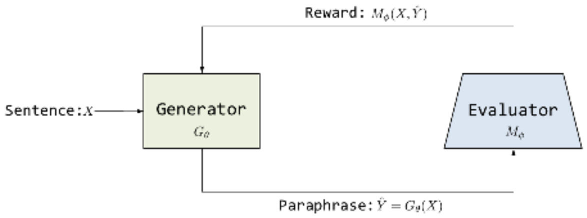

Reward: $M_\phi(X,\hat{Y})$
Sentence: X
Generator
G_θ
Evaluator
M_ϕ
Paraphrase: $\hat{Y} = G_\theta(X)$

Figure 1: Framework of RbM (Reinforced by Matching).

Section 2 defines the models of generator and evaluator. In section 3, we formalize the problem of learning the models of generator and evaluator. In section 4, we report our experimental results. In section 5, we introduce related work.

2 Models

This section explains our framework for paraphrase generation, containing two models, the generator and evaluator.

2.1 Problem and Framework

Given an input sequence of words $X = [x_1, \ldots, x_S]$ with length S, we aim to generate an output sequence of words $Y = [y_1, \ldots, y_T]$ with length T that has the same meaning as X. We denote the pair of sentences in paraphrasing as (X, Y). We use $Y_{1:t}$ to denote the subsequence of Y ranging from 1 to t and use $\hat{Y}$ to denote the sequence generated by a model.

We propose a framework, which contains a generator and an evaluator, called **RbM** (Reinforced by Matching). Specifically, for the generator we adopt the Seq2Seq architecture with attention and copy mechanism (Bahdanau et al., 2015; See et al., 2017), and for the evaluator we adopt the decomposable attention-based deep matching model (Parikh et al., 2016). We denote the generator as G_θ and the evaluator as M_ϕ, where θ and ϕ represent their parameters respectively. Figure 1 gives an overview of our framework. Basically the generator can generate a paraphrase of a given sentence and the evaluator can judge how semantically similar the two sentences are.

2.2 Generator: Seq2Seq Model

In this work, paraphrase generation is defined as a sequence-to-sequence (Seq2Seq) learning problem. Given input sentence X, the goal is to learn a model G_θ that can generate a sentence

$\hat{Y} = G_\theta(X)$ as its paraphrase. We choose the *pointer-generator* proposed by See et al. (2017) as the generator. The model is built based on the encoder-decoder framework (Cho et al., 2014; Sutskever et al., 2014), both of which are implemented as recurrent neural networks (RNN). The encoder RNN transforms the input sequence X into a sequence of hidden states $H = [h_1, \dots, h_S]$. The decoder RNN generates an output sentence Y on the basis of the hidden states. Specifically it predicts the next word at each position by sampling from $\hat{y}_t \sim p(y_t|Y_{1:t-1}, X) = g(s_t, c_t, y_{t-1})$, where s_t is the decoder state, c_t is the context vector, y_{t-1} is the previous word, and g is a feed-forward neural network. Attention mechanism (Bahdanau et al., 2015) is introduced to compute the context vector as the weighted sum of encoder states:

$$c_t = \sum_{i=1}^{S} \alpha_{ti} h_i, \quad \alpha_{ti} = \frac{\exp \eta(s_{t-1}, h_i)}{\sum_{j=1}^{S} \exp \eta(s_{t-1}, h_j)},$$

where α_{ti} represents the attention weight and η is the *attention function*, which is a feed-forward neural network.

Paraphrasing often needs copying words from the input sentence, for instance, named entities. The pointer-generator model allows either generating words from a vocabulary or copying words from the input sequence. Specifically the probability of generating the next word is given by a mixture model:

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

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

2.3 Evaluator: Deep Matching Model

In this work, paraphrase evaluation (identification) is casted as a problem of learning of sentence matching. The goal is to learn a real-valued function $M_\phi(X, Y)$ that can represent the matching degree between the two sentences as paraphrases of each other. A variety of learning techniques have been developed for matching sentences, from linear models (e.g., Wu et al. (2013)) to neural network based models (e.g., Socher et al. (2011); Hu et al. (2014)). We choose a simple yet effective neural network architecture, called the *decomposable-attention* model (Parikh et al., 2016), as the evaluator. The evaluator can calculate the semantic similarity between two sentences:

$$M_\phi(X, Y) = H(\sum_{i=1}^{S} G([e(x_i), \bar{x}_i]), \sum_{j=1}^{T} G([e(y_j), \bar{y}_j])),$$

where $e(\cdot)$ denotes a word embedding, $\bar{x}_i$ and $\bar{y}_j$ denote inter-attended vectors, H and G are feed-forward networks. We refer the reader to Parikh et al. (2016) for details. In addition, we add *positional encodings* to the word embedding vectors to incorporate the order information of the words, following the idea in Vaswani et al. (2017).

3 Learning

This section explains how to learn the generator and evaluator using deep reinforcement learning.

3.1 Learning of Generator

Given training data (X, Y), the generator G_θ is first trained to maximize the conditional log likelihood (negative cross entropy):

$$\mathcal{L}_{\text{Seq2Seq}}(\theta) = \sum\nolimits_{t=1}^{T} \log p_\theta(y_t|Y_{1:t-1}, X). \quad (1)$$

When computing the conditional probability of the next word as above, we choose the previous word y_{t-1} in the ground-truth rather than the word $\hat{y}_{t-1}$ generated by the model. This technique is called *teacher forcing*.

With teacher forcing, the discrepancy between training and prediction (also referred to as *exposure bias*) can quickly accumulate errors along the generated sequence (Bengio et al., 2015; Ranzato et al., 2016). Therefore, the generator G_θ is next fine-tuned using RL, where the reward is given by the evaluator.

In the RL formulation, generation of the next word represents an *action*, the previous words represent a *state*, and the probability of generation $p_\theta(y_t|Y_{1:t-1}, X)$ induces a stochastic *policy*. Let r_t denote the *reward* at position t. The goal of RL is to find a policy (i.e., a generator) that maximizes the expected cumulative reward:

$$\mathcal{L}_{RL}(\theta) = \mathbb{E}_{p_\theta(\hat{Y}|X)} \sum_{t=1}^{T} r_t(X, \hat{Y}_{1:t}). \quad (2)$$

We define a positive reward at the end of sequence ($r_T = R$) and a zero reward at the other positions. The reward R is given by the evaluator M_ϕ. In particular, when a pair of input sentence X and generated paraphrase $\hat{Y} = G_\theta(X)$ is given, the reward is calculated by the evaluator $R = M_\phi(X, \hat{Y})$.

We can then learn the optimal policy by employing policy gradient. According to the policy gradient theorem (Williams, 1992; Sutton et al., 2000), the gradient of the expected cumulative reward can be calculated by

$$\nabla_\theta \mathcal{L}_{RL}(\theta) = \sum_{t=1}^{T} [\nabla_\theta \log p_\theta(\hat{y}_t | \hat{Y}_{1:t-1}, X)] r_t. \tag{3}$$

The generator can thus be learned with stochastic gradient descent methods such as Adam (Kingma and Ba, 2015).

3.2 Learning of Evaluator

The evaluator works as the reward function in RL of the generator and thus is essential for the task. We propose two methods for learning the evaluator in different settings. When there are both positive and negative examples of paraphrases, the evaluator is trained by *supervised learning* (SL). When only positive examples are available (usually the same data as the training data of the generator), the evaluator is trained by *inverse reinforcement learning* (IRL).

Supervised Learning

Given a set of positive and negative examples (paraphrase pairs), we conduct supervised learning of the evaluator with the pointwise cross entropy loss:

$$\mathcal{J}_{\text{SL}}(\phi) = -\log M_\phi(X, Y) - \log(1 - M_\phi(X, Y^-)), \tag{4}$$

where Y^- represents a sentence that is not a paraphrase of X. The evaluator M_ϕ here is defined as a classifier, trained to distinguish negative example (X, Y^-) from positive example (X, Y).

We call this method **RbM-SL** (Reinforced by Matching with Supervised Learning). The evaluator M_ϕ trained by supervised learning can make a judgement on whether two sentences are paraphrases of each other. With a well-trained evaluator M_ϕ, we further train the generator G_θ by reinforcement learning using M_ϕ as a reward function. Figure 2a shows the learning process of RbM-SL. The detailed training procedure is shown in Algorithm 1 in Appendix A.

Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is a sub-problem of reinforcement learning (RL), about learning of a reward function given *expert demonstrations*, which are sequences of states and actions from an expert (optimal) policy. More specifically, the goal is to find an optimal reward function R^* with which the expert policy $p_{\theta^*}(Y|X)$ really becomes optimal among all possible policies, i.e.,

$$\mathbb{E}_{p_{\theta^*}(Y|X)} R^*(Y) \geq \mathbb{E}_{p_\theta(\hat{Y}|X)} R^*(\hat{Y}), \quad \forall \theta.$$

In the current problem setting, the problem becomes learning of an optimal reward function (evaluator) given a number of paraphrase pairs given by human experts (expert demonstrations).

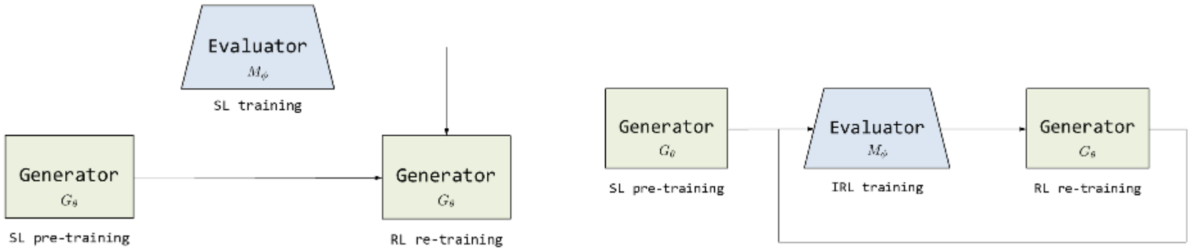
To learn an optimal reward (matching) function is challenging, because the expert demonstrations might not be optimal and the reward function might not be rigorously defined. To deal with the problem, we employ the maximum margin formulation of IRL inspired by Ratliff et al. (2006).

The maximum margin approach ensures the learned reward function has the following two desirable properties in the paraphrase generation task: (a) given the same input sentence, a reference from humans should have a higher reward than the ones generated by the model; (b) the margins between the rewards should become smaller when the paraphrases generated by the model get closer to a reference given by humans. We thus specifically consider the following optimization problem for learning of the evaluator:

$$\mathcal{J}_{\text{IRL}}(\phi) = \max(0, 1 - \zeta + M_\phi(X, \hat{Y}) - M_\phi(X, Y)), \tag{5}$$

where ζ is a slack variable to measure the agreement between $\hat{Y}$ and Y. In practice we set ζ = ROUGE-L$(\hat{Y}, Y)$. Different from RbM-SL, the evaluator M_ϕ here is defined as a ranking model that assigns higher rewards to more plausible paraphrases.

Once the reward function (evaluator) is learned, it is then used to improve the policy function (generator) through policy gradient. In fact, the generator G_θ and the evaluator M_ϕ are trained alternatively. We call this method **RbM-IRL** (Reinforced by Matching with Inverse Reinforcement Learning). Figure 2b shows the learning process of RbM-IRL. The detailed training procedure is shown in Algorithm 2 in Appendix A.



(a) RbM-SL

(b) RbM-IRL

Figure 2: Learning Process of RbM models: (a) RbM-SL, (b) RbM-IRL.

We can formalize the whole learning procedure as the following optimization problem:

$$\min_{\phi} \max_{\theta} \mathbb{E}_{p_\theta(\hat{Y}|X)} \mathcal{J}_{\text{IRL}}(\phi). \tag{6}$$

RbM-IRL can make effective use of sequences generated by the generator for training of the evaluator. As the generated sentences become closer to the ground-truth, the evaluator also becomes more discriminative in identifying paraphrases.

It should be also noted that for both RbM-SL and RbM-IRL, once the evaluator is learned, the reinforcement learning of the generator only needs *non-parallel sentences* as input. This makes it possible to further train the generator and enhance the generalization ability of the generator.

3.3 Training Techniques

Reward Shaping

In the original RL of the generator, only a positive reward R is given at the end of sentence. This provides sparse supervision signals and can make the model greatly degenerate. Inspired by the idea of reward shaping (Ng et al., 1999; Bahdanau et al., 2017), we estimate the intermediate cumulative reward (value function) for each position, that is

$$Q_t = \mathbb{E}_{p_\theta(Y_{t+1:T}|\hat{Y}_{1:t},X)} R(X, [\hat{Y}_{1:t}, Y_{t+1:T}]),$$

by Monte-Carlo simulation, in the same way as in Yu et al. (2017):

$$Q_t = \begin{cases} \frac{1}{N}\sum_{n=1}^{n=N} M_\phi(X, [\hat{Y}_{1:t}, \hat{Y}^n_{t+1:T}]), & t < T \\ M_\phi(X, \hat{Y}), & t = T, \end{cases} \tag{7}$$

where N is the sample size and $\hat{Y}^n_{t+1:T} \sim p_\theta(Y_{t+1:T}|\hat{Y}_{1:t}, X)$ denotes simulated subsequences randomly sampled starting from the $(t+1)$-th word. During training of the generator, the reward r_t in policy gradient (3) is replaced by Q_t estimated in (7).

Reward Rescaling

In practice, RL algorithms often suffer from instability in training. A common approach to reduce the variance is to subtract a baseline reward from the value function. For instance, a simple baseline can be a moving average of historical rewards. While in RbM-IRL, the evaluator keeps updating during training. Thus, keeping track of a baseline reward is unstable and inefficient. Inspired by Guo et al. (2018), we propose an efficient reward rescaling method based on ranking. For a batch of D generated paraphrases $\{\hat{Y}^d\}_{d=1}^D$, each associated with a reward $R^d = M_\phi(X^d, \hat{Y}^d)$, we rescale the rewards by

$$\bar{R}^d = \sigma(\delta_1 \cdot (0.5 - \frac{\text{rank}(d)}{D})) - 0.5, \tag{8}$$

where $\sigma(\cdot)$ is the sigmoid function, rank(d) is the rank of R^d in $\{R^1, ..., R^D\}$, and δ_1 is a scalar controlling the variance of rewards. A similar strategy is applied into estimation of in-sequence value function for each word, and the final rescaled value function is

$$\bar{Q}^d_t = \sigma(\delta_2 \cdot (0.5 - \frac{\text{rank}(t)}{T})) - 0.5 + \bar{R}^d, \tag{9}$$

where rank(t) is the rank of Q^d_t in $\{Q^d_1, ..., Q^d_T\}$.

Reward rescaling has two advantages. First, the mean and variance of $\bar{Q}^d_t$ are controlled and hence they make the policy gradient more stable, even with a varying reward function. Second, when the evaluator M_ϕ is trained with the ranking loss as in RbM-IRL, it is better to inform which paraphrase is better, rather than to provide a scalar reward in a range. In our experiment, we find that this method can bring substantial gains for RbM-SL and RbM-IRL, but not for RL with ROUGE as reward.

Curriculum Learning

RbM-IRL may not achieve its best performance if all of the training instances are included in training at the beginning. We employ a curriculum learning strategy (Bengio et al., 2009) for it. During the training of the evaluator M_ϕ, each example k is associated with a weight w^k, i.e.

$$\mathcal{J}^k_{\text{IRL-CL}}(\phi) = w^k \max(0,1 - \zeta^k + M_\phi(X^k, \hat{Y}^k) - M_\phi(X^k, Y^k)) \quad (10)$$

In curriculum learning, w^k is determined by the difficulty of the example. At the beginning, the training procedure concentrates on relatively simple examples, and gradually puts more weights on difficult ones. In our case, we use the edit distance $\mathcal{E}(X, Y)$ between X and Y as the measure of difficulty for paraphrasing. Specifically, w^k is determined by $w^k \sim \text{Binomial}(p^k, 1)$, and $p^k = \sigma(\delta_3 \cdot (0.5 - \frac{\text{rank}(\mathcal{E}(X^k,Y^k))}{K}))$, where K denotes the batch size for training the evaluator. For δ_3, we start with a relatively high value and gradually decrease it. In the end each example will be sampled with a probability around 0.5. In this manner, the evaluator first learns to identify paraphrases with small modifications on the input sentences (e.g. "*what 's*" and "*what is*"). Along with training it gradually learns to handle more complicated paraphrases (e.g. "*how can I*" and "*what is the best way to*").

4 Experiment

4.1 Baselines and Evaluation Measures

To compare our methods (RbM-SL and RbM-IRL) with existing neural network based methods, we choose five baseline models: the attentive Seq2Seq model (Bahdanau et al., 2015), the stacked Residual LSTM networks (Prakash et al., 2016), the variational auto-encoder (VAE-SVG-eq) (Gupta et al., 2018) [1], the pointer-generator (See et al., 2017), and the reinforced pointer-generator with ROUGE-2 as reward (RL-ROUGE) (Ranzato et al., 2016).

We conduct both automatic and manual evaluation on the models. For the automatic evaluation, we adopt four evaluation measures: ROUGE-1, ROUGE-2 (Lin, 2004), BLEU (Papineni et al., 2002) (up to at most bi-grams) and METEOR (Lavie and Agarwal, 2007). As pointed out, ideally it would be better not to merely use a lexical measure like ROUGE or BLEU for evaluation of paraphrasing. We choose to use them for reproducibility of our experimental results by others. For the manual evaluation, we conduct evaluation on the generated paraphrases in terms of relevance and fluency.

4.2 Datasets

We evaluate our methods with the **Quora** question pair dataset [2] and **Twitter** URL paraphrasing corpus (Lan et al., 2017). Both datasets contain positive and negative examples of paraphrases so that we can evaluate the RbM-SL and RbM-IRL methods. We randomly split the Quora dataset in two different ways obtaining two experimental settings: Quora-I and Quora-II. In Quora-I, we partition the dataset by question pairs, while in Quora-II, we partition by question ids such that there is no shared question between the training and test/validation datasets. In addition, we sample a smaller training set in Quora-II to make the task more challenging. Twitter URL paraphrasing corpus contains two subsets, one is labeled by human annotators while the other is labeled automatically by algorithm. We sample the test and validation set from the labeled subset, while using the remaining pairs as training set. For RbM-SL, we use the labeled subset to train the evaluator M_ϕ. Compared to Quora-I, it is more difficult to achieve a high performance with Quora-II. The Twitter corpus is even more challenging since the data contains more noise. The basic statistics of datasets are shown in Table 1.

Table 1: Statistics of datasets.

	Generator			Evaluator (RbM-SL)	
Dataset	#Train	#Test	#Validation	#Positive	#Negative
Quora-I	100K	30K	3K	100K	160K
Quora-II	50K	30K	3K	50K	160K
Twitter	110K	5K	1K	10K	40K

4.3 Implementation Details

Generator We maintain a fixed-size vocabulary of 5K shared by the words in input and output, and truncate all the sentences longer than 20 words. The model architecture, word embedding size and LSTM cell size are as the same as reported in See et al. (2017). We use Adadgrad optimizer (Duchi et al., 2011) in the supervised pre-training and Adam optimizer in the reinforcement learning, with the batch size of 80. We also fine-tune the

[1] We directly present the results reported in Gupta et al. (2018) on the same dateset and settings.

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

Table 2: Performances on Quora datasets.

	Quora-I				Quora-II			
Models	ROUGE-1	ROUGE-2	BLEU	METEOR	ROUGE-1	ROUGE-2	BLEU	METEOR
Seq2Seq	58.77	31.47	36.55	26.28	47.22	20.72	26.06	20.35
Residual LSTM	59.21	32.43	37.38	28.17	48.55	22.48	27.32	22.37
VAE-SVG-eq	-	-	-	25.50	-	-	-	22.20
Pointer-generator	61.96	36.07	40.55	30.21	51.98	25.16	30.01	24.31
RL-ROUGE	63.35	37.33	41.83	30.96	54.50	27.50	32.54	25.67
RbM-SL (ours)	**64.39**	**38.11**	**43.54**	**32.84**	**57.34**	**31.09**	**35.81**	**28.12**
RbM-IRL (ours)	64.02	37.72	43.09	31.97	56.86	29.90	34.79	26.67

Table 3: Performances on Twitter corpus.

	Twitter			
Models	ROUGE-1	ROUGE-2	BLEU	METEOR
Seq2Seq	30.43	14.61	30.54	12.80
Residual LSTM	32.50	16.86	33.90	13.65
Pointer-generator	38.31	21.22	40.37	17.62
RL-ROUGE	40.16	22.99	42.73	18.89
RbM-SL (ours)	41.87	24.23	44.67	19.97
RbM-IRL (ours)	**42.15**	**24.73**	**45.74**	**20.18**

Table 4: Human evaluation on Quora datasets.

	Quora-I		Quora-II	
Models	Relevance	Fluency	Relevance	Fluency
Pointer-generator	3.23	4.55	2.34	2.96
RL-ROUGE	3.56	4.61	2.58	3.14
RbM-SL (ours)	**4.08**	4.67	**3.20**	3.48
RbM-IRL (ours)	4.07	**4.69**	2.80	**3.53**
Reference	4.69	4.95	4.68	4.90

Seq2Seq baseline models with Adam optimizer for a fair comparison. 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. During the RL training, the learning rate decreases to 1e-5 and the size of Monte-Carlo sample is 4. To make the training more stable, we use the ground-truth with reward of 0.1.

Evaluator We use the pretrained GoogleNews 300-dimension word vectors [3] in Quora dataset and 200-dimension GloVe word vectors [4] in Twitter corpus. Other model settings are the same as in Parikh et al. (2016). For evaluator in RbM-SL we set the learning rate as 0.05 and the batch size as 32. For the evaluator of M_ϕ in RbM-IRL, the learning rate decreases to 1e-2, and we use the batch size of 80.

We use the technique of reward rescaling as mentioned in section 3.3 in training RbM-SL and RbM-IRL. In RbM-SL, we set δ_1 as 12 and δ_2 as 1. In RbM-IRL, we keep δ_2 as 1 all the time and decrease δ_1 from 12 to 3 and δ_3 from 15 to 8 during curriculum learning. In ROUGE-RL, we take the exponential moving average of historical rewards as baseline reward to stabilize the training:

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

where b_m is the baseline b at iteration m, $\overline{Q}$ is the mean value of reward, and we set λ as 0.1 by grid search.

4.4 Results and Analysis

Automatic evaluation Table 2 shows the performances of the models on Quora datasets. In both settings, we find that the proposed RbM-SL and RbM-IRL models outperform the baseline models in terms of all the evaluation measures. Particularly in Quora-II, RbM-SL and RbM-IRL make significant improvements over the baselines, which demonstrates their higher ability in learning for paraphrase generation. On Quora dataset, RbM-SL is constantly better than RbM-IRL for all the automatic measures, which is reasonable because RbM-SL makes use of additional labeled data to train the evaluator. Quora datasets contains a large number of high-quality non-paraphrases, i.e., they are literally similar but semantically different, for instance "*are analogue clocks better than digital*" and "*is analogue better than digital*". Trained with the data, the evaluator tends to become more capable in paraphrase identification. With additional evaluation on Quora data, the evaluator used in RbM-SL can achieve an accuracy of 87% on identifying positive and negative pairs of paraphrases.

Table 3 shows the performances on the Twitter corpus. Our models again outperform the baselines in terms of all the evaluation measures. Note that RbM-IRL performs better than RbM-SL in this case. The reason might be that the evaluator

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

[4] https://nlp.stanford.edu/projects/glove/

of RbM-SL might not be effectively trained with the relatively small dataset, while RbM-IRL can leverage its advantage in learning of the evaluator with less data.

In our experiments, we find that the training techniques proposed in section 3.3 are all necessary and effective. Reward shaping is by default employed by all the RL based models. Reward rescaling works particularly well for the RbM models, where the reward functions are learned from data. Without reward rescaling, RbM-SL can still outperform the baselines but with smaller margins. For RbM-IRL, curriculum learning is necessary for its best performance. Without curriculum learning, RbM-IRL only has comparable performance with ROUGE-RL.

Human evaluation We randomly select 300 sentences from the test data as input and generate paraphrases using different models. The pairs of paraphrases are then aggregated and partitioned into seven random buckets for seven human assessors to evaluate. The assessors are asked to rate each sentence pair according to the following two criteria: *relevance* (the paraphrase sentence is semantically close to the original 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 ranging from 1 to 5. To reduce the evaluation variance, there is a detailed evaluation guideline for the assessors in Appendix B. Each paraphrase is rated by two assessors, and then averaged as the final judgement. The agreement between assessors is moderate (kappa=0.44).

Table 4 demonstrates the average ratings for each model, including the ground-truth references. Our models of RbM-SL and RbM-IRL get better scores in terms of relevance and fluency than the baseline models, and their differences are statistically significant (paired t-test, p-value $<$ 0.01). We note that in human evaluation, RbM-SL achieves the best relevance score while RbM-IRL achieves the best fluency score.

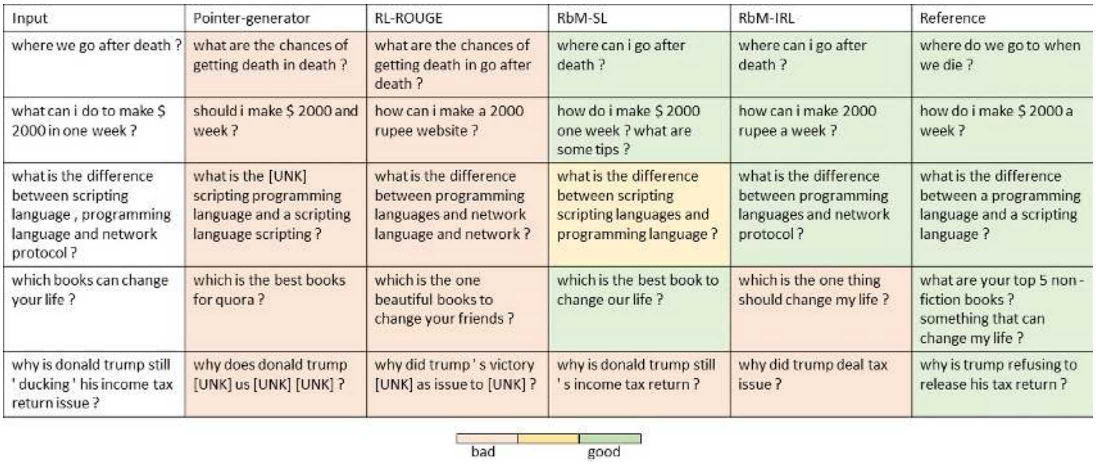
Case study Figure 3 gives some examples of generated paraphrases by the models on Quora-II for illustration. The first and second examples show the superior performances of RbM-SL and RbM-IRL over the other models. In the third example, both RbM-SL and RbM-IRL capture accurate paraphrasing patterns, while the other models wrongly segment and copy words from the input sentence. Compared to RbM-SL with an error of repeating the word *scripting*, RbM-IRL generates a more fluent paraphrase. The reason is that the evaluator in RbM-IRL is more capable of measuring the fluency of a sentence. In the fourth example, RL-ROUGE generates a totally non-sense sentence, and pointer-generator and RbM-IRL just cover half of the content of the original sentence, while RbM-SL successfully rephrases and preserves all the meaning. All of the models fail in the last example, because the word *ducking* is a rare word that never appears in the training data. Pointer-generator and RL-ROUGE generate totally irrelevant words such as *UNK* token or *victory*, while RbM-SL and RbM-IRL still generate topic-relevant words.

5 Related Work

Neural paraphrase generation recently draws attention in different application scenarios. The task is often formalized as a sequence-to-sequence (Seq2Seq) learning problem. Prakash et al. (2016) employ a stacked residual LSTM network in the Seq2Seq model to enlarge the model capacity. Cao et al. (2017) utilize an additional vocabulary to restrict word candidates during generation. Gupta et al. (2018) use a variational auto-encoder framework to generate more diverse paraphrases. Ma et al. (2018) utilize an attention layer instead of a linear mapping in the decoder to pick up word candidates. Iyyer et al. (2018) harness syntactic information for controllable paraphrase generation. Zhang and Lapata (2017) tackle a similar task of sentence simplification withe Seq2Seq model coupled with deep reinforcement learning, in which the reward function is manually defined for the task. Similar to these works, we also pretrain the paraphrase generator within the Seq2Seq framework. The main difference lies in that we use another trainable neural network, referred to as evaluator, to guide the training of the generator through reinforcement learning.

There is also work on paraphrasing generation in different settings. For example, Mallinson et al. (2017) leverage bilingual data to produce paraphrases by pivoting over a shared translation in another language. Wieting et al. (2017); Wieting and Gimpel (2018) use neural machine translation to generate paraphrases via back-translation of bilingual sentence pairs. Buck et al. (2018) and Dong et al. (2017) tackle the problem of QA-specific paraphrasing with the guidance from an external

Figure 3: Examples of the generated paraphrases by different models on Quora-II.



Input	Pointer-generator	RL-ROUGE	RbM-SL	RbM-IRL	Reference
where we go after death ?	what are the chances of getting death in death ?	what are the chances of getting death in go after death ?	where can i go after death ?	where can i go after death ?	where do we go to when we die ?
what can i do to make $ 2000 in one week ?	should i make $ 2000 and week ?	how can i make a 2000 rupee website ?	how do i make $ 2000 one week ? what are some tips ?	how can i make 2000 rupee a week ?	how do i make $ 2000 a week ?
what is the difference between scripting language , programming language and network protocol ?	what is the [UNK] scripting programming language and a scripting language scripting ?	what is the difference between programming languages and network language and network ?	what is the difference between scripting scripting languages and programming language ?	what is the difference between programming languages and network protocol ?	what is the difference between a programming language and a scripting language ?
which books can change your life ?	which is the best books for quora ?	which is the one beautiful books to change your friends ?	which is the best book to change our life ?	which is the one thing should change my life ?	what are your top 5 non - fiction books ? something that can change my life ?
why is donald trump still ' ducking ' his income tax return issue ?	why does donald trump [UNK] us [UNK] [UNK] ?	why did trump ' s victory [UNK] as issue to [UNK] ?	why is donald trump still ' s income tax return ?	why did trump deal tax issue ?	why is trump refusing to release his tax return ?

QA system and an associated evaluation metric.

Inverse reinforcement learning (IRL) aims to learn a reward function from expert demonstrations. Abbeel and Ng (2004) propose *apprenticeship learning*, which uses a feature based linear reward function and learns to match feature expectations. Ratliff et al. (2006) cast the problem as structured maximum margin prediction. Ziebart et al. (2008) propose max entropy IRL in order to solve the problem of expert suboptimality. Recent work involving deep learning in IRL includes Finn et al. (2016b) and Ho et al. (2016). There does not seem to be much work on IRL for NLP. In Neu and Szepesvári (2009), parsing is formalized as a feature expectation matching problem. Wang et al. (2018) apply adversarial inverse reinforcement learning in visual story telling. To the best of our knowledge, our work is the first that applies deep IRL into a Seq2Seq task.

Generative Adversarial Networks (GAN) (Goodfellow et al., 2014) is a family of unsupervised generative models. GAN contains a generator and a discriminator, respectively for generating examples from random noises and distinguishing generated examples from real examples, and they are trained in an adversarial way. There are applications of GAN on NLP, such as text generation (Yu et al., 2017; Guo et al., 2018) and dialogue generation (Li et al., 2017). RankGAN (Lin et al., 2017) is the one most similar to RbM-IRL that employs a ranking model as the discriminator. However, RankGAN works for text generation rather than sequence-to-sequence learning, and training of generator in RankGAN relies on parallel data while the training of RbM-IRL can use non-parallel data.

There are connections between GAN and IRL as pointed by Finn et al. (2016a); Ho and Ermon (2016). However, there are significant differences between GAN and our RbM-IRL model. GAN employs the discriminator to distinguish generated examples from real examples, while RbM-IRL employs the evaluator as a reward function in RL. The generator in GAN is trained to maximize the loss of the discriminator in an adversarial way, while the generator in RbM-IRL is trained to maximize the expected cumulative reward from the evaluator.

6 Conclusion

In this paper, we have proposed a novel deep reinforcement learning approach to paraphrase generation, with a new framework consisting of a generator and an evaluator, modeled as sequence-to-sequence learning model and deep matching model respectively. The generator, which is for paraphrase generation, is first trained via sequence-to-sequence learning. The evaluator, which is for paraphrase identification, is then trained via supervised learning or inverse reinforcement learning in different settings. With a well-trained evaluator, the generator is further fine-tuned by reinforcement learning to produce more accurate paraphrases. The experiment results demonstrate that the proposed method can significantly improve the quality of paraphrase generation upon the baseline methods. In the future, we plan to apply the framework and training techniques into other tasks, such as machine translation and dialogue.

Acknowledgments

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

References

Pieter Abbeel and Andrew Y Ng. 2004. Apprenticeship learning via inverse reinforcement learning. In *ICML*.

Dzmitry Bahdanau, Philemon Brakel, Kelvin Xu, Anirudh Goyal, Ryan Lowe, Joelle Pineau, Aaron Courville, and Yoshua Bengio. 2017. An actor-critic algorithm for sequence prediction. In *ICLR*.

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

Samy Bengio, Oriol Vinyals, Navdeep Jaitly, and Noam Shazeer. 2015. Scheduled sampling for sequence prediction with recurrent neural networks. In *NIPS*.

Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. 2009. Curriculum learning. In *ICML*.

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

Christian Buck, Jannis Bulian, Massimiliano Ciaramita, Andrea Gesmundo, Neil Houlsby, Wojciech Gajewski, and Wei Wang. 2018. Ask the right questions: Active question reformulation with reinforcement learning. In *ICLR*.

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

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

Li Dong, Jonathan Mallinson, Siva Reddy, and Mirella Lapata. 2017. Learning to paraphrase for question answering. In *EMNLP*.

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.

Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. 2016a. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *NIPS 2016 Workshop on Adversarial Training*.

Chelsea Finn, Sergey Levine, and Pieter Abbeel. 2016b. Guided cost learning: Deep inverse optimal control via policy optimization. In *ICML*.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *NIPS*.

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

Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. 2018. A deep generative framework for paraphrase generation. In *AAAI*.

Jonathan Ho and Stefano Ermon. 2016. Generative adversarial imitation learning. In *NIPS*.

Jonathan Ho, Jayesh Gupta, and Stefano Ermon. 2016. Model-free imitation learning with policy optimization. In *ICML*.

Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. 2014. Convolutional neural network architectures for matching natural language sentences. In *NIPS*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *NAACL*.

David Kauchak and Regina Barzilay. 2006. Paraphrasing for automatic evaluation. In *NAACL*.

Diederik Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *ICLR*.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. In *EMNLP*.

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*.

Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. 2017. Adversarial learning for neural dialogue generation. In *EMNLP*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *ACL-04 workshop*.

Kevin Lin, Dianqi Li, Xiaodong He, Zhengyou Zhang, and Ming-Ting Sun. 2017. Adversarial ranking for language generation. In *NIPS*.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018. Word embedding attention network: Generating words by querying distributed word representations for paraphrase generation. In *NAACL*.

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *EACL*.

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

Shashi Narayan, Siva Reddy, and Shay B Cohen. 2016. Paraphrase generation from latent-variable pcfgs for semantic parsing. In *INLG*.

Gergely Neu and Csaba Szepesvári. 2009. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2):303–337.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*.

Ankur P Parikh, Oscar Täckström, Dipanjan Das, and Jakob Uszkoreit. 2016. A decomposable attention model for natural language inference. In *EMNLP*.

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. In *COLING*.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *EMNLP*.

Marc'Aurelio Ranzato, Sumit Chopra, Michael Auli, and Wojciech Zaremba. 2016. Sequence level training with recurrent neural networks. In *ICLR*.

Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. 2006. Maximum margin planning. In *ICML*.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *EMNLP*.

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

Lifeng Shang, Zhengdong Lu, and Hang Li. 2015. Neural responding machine for short-text conversation. In *ACL*.

Richard Socher, Eric H Huang, Jeffrey Pennin, Christopher D Manning, and Andrew Y Ng. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *EMNLP*.

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

Richard S Sutton, David A McAllester, Satinder P Singh, and Yishay Mansour. 2000. Policy gradient methods for reinforcement learning with function approximation. In *NIPS*.

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

Oriol Vinyals and Quoc Le. 2015. A neural conversational model.

Xin Wang, Wenhu Chen, Yuan-Fang Wang, and William Yang Wang. 2018. No metrics are perfect: Adversarial reward learning for visual storytelling. In *ACL*.

John Wieting and Kevin Gimpel. 2018. Paranmt-50m: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning paraphrastic sentence embeddings from back-translated bitext. In *EMNLP*.

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

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

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

Jun Yin, Xin Jiang, Zhengdong Lu, Lifeng Shang, Hang Li, and Xiaoming Li. 2016. Neural generative question answering. In *IJCAI*.

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

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *EMNLP*.

Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. Application-driven statistical paraphrase generation. In *ACL*.

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

Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. 2008. Maximum entropy inverse reinforcement learning. In *AAAI*.

A Algorithms of RbM-SL and RbM-IRL

Algorithm 1: Training Procedure of RbM-SL

```
Input  : A corpus of paraphrase pairs $\{(X,Y)\}$, a corpus of
         non-paraphrase pairs $\{(X,Y^-)\}$, a corpus of (non-parallel)
         sentences $\{X\}$.
Output: Generator $G_{\theta'}$
Train the evaluator $M_\phi$ with $\{(X,Y)\}$ and $\{(X,Y^-)\}$;
Pre-train the generator $G_\theta$ with $\{(X,Y)\}$;
Init $G_{\theta'} := G_\theta$;
while not converge do
    Sample a sentence $X = [x_1, \dots, x_S]$ from the paraphrase corpus or the
      non-parallel corpus;
    Generate a sentence $\hat{Y} = [\hat{y}_1, \dots, \hat{y}_T]$ according to $G_{\theta'}$ given input $X$;
    Set the gradient $g_{\theta'} = 0$;
    for $t = 1$ to $T$ do
        Run $N$ Monte Carlo simulations:
          $\{\widehat{Y}^1_{t+1:T}, ...\widehat{Y}^N_{t+1:T}\} \sim p_{\theta'}(Y_{t+1:T}|\hat{Y}_{1:t}, X)$;
        Compute the value function by
          $Q_t = \begin{cases} \frac{1}{N}\sum_{n=1}^{N} M_\phi(X, [\hat{Y}_{1:t}, \widehat{Y}^n_{t+1:T}]), & t < T \\ M_\phi(X, \hat{Y}), & t = T. \end{cases}$
          Rescale the reward to $\bar{Q}_t$ by (8);
        Accumulate $\theta'$-gradient: $g_{\theta'} := g_{\theta'} + \nabla_\theta \log p_{\theta'}(\hat{y}_t|\hat{Y}_{1:t-1}, X)\bar{Q}_t$
    end
    Update $G_{\theta'}$ using the gradient $g_{\theta'}$ with
      learning rate $\gamma_G$: $G_{\theta'} := G_{\theta'} + \gamma_G g_{\theta'}$
end
Return $G_{\theta'}$
```

Algorithm 2: Training Procedure of RbM-IRL

```
Input  : A corpus of paraphrase pairs $\{(X,Y)\}$, a corpus of
         (non-parallel) sentences $\{X\}$.
Output: Generator $G_{\theta'}$, evaluator $M_{\phi'}$
Pre-train the generator $G_\theta$ with $\{(X,Y)\}$;
Init $G_{\theta'} := G_\theta$ and $M_{\phi'}$;
while not converge do
    while not converge do
        Sample a sentence $X = [x_1, \dots, x_S]$ from the paraphrase corpus;
        Generate a sentence $\hat{Y} = [\hat{y}_1, \dots, \hat{y}_T]$ according to $G_{\theta'}$
          given input $X$;
        Calculate $\phi'$-gradient:
          $g_{\phi'} := \nabla_\phi \mathcal{J}_{\text{IRL-CL}}(\phi)$;
        Update $M_{\phi'}$ using the gradient $g_{\phi'}$
          with learning rate $\gamma_M$:
          $M_{\phi'} := M_{\phi'} - \gamma_M g_{\phi'}$
    end
    Train $G_{\theta'}$ with $M_{\phi'}$ as in Algorithm 1;
end
Return $G_{\theta'}$, $M_{\phi'}$
```

B Human Evaluation Guideline

Please judge the paraphrases from the following two criteria:

(1) **Grammar and Fluency**: the paraphrase is acceptable as natural language text, and the grammar is correct;

(2) **Coherent and Consistent**: please view from the perspective of the original poster, to what extent the answer of paraphrase is helpful for you with respect to the original question. Specifically, you can consider following aspects:

   - Relatedness: it should be topically relevant to the original question.
   - Type of question: the type of the original question remains the same in paraphrase.
   - Informative: no information loss in paraphrase.

For each paraphrase, give two separate score ranking from 1 to 5. The meaning of specific score is as following:

- Grammar and Fluency
  - 5: Without any grammatical error;
  - 4: Fluent and has one minor grammatical error that does not affect understanding, e.g. *what is the best ways to learn programming*;
  - 3: Basically fluent and has two or more minor grammatical errors or one serious grammatical error that does not have strong impact on understanding, e.g. *what some good book for read*;
  - 2: Can not understand what it means but it is still in the form of human language, e.g. *what is the best movie of movie*;
  - 1: Non-sense composition of words and not in the form of human language, e.g. *how world war iii world war.*
- Coherent and Consistent
  - 5: Accurate paraphrase with exact the same meaning of the source sentence;
  - 4: Basically the same meaning of the source sentence but does not cover some minor content, e.g. *what are some good places to visit in hong kong during summer → can you suggest some places to visit in hong kong*;
  - 3: Cover part of the content of source sentence and has serious information loss, e.g. *what is the best love movie by wong ka wai → what is the best movie*;
  - 2: Topic relevant but fail to cover most of the content of source sentence, e.g. *what is some tips to learn english → when do you start to learn english*;
  - 1: Topic irrelevant or even can not understand what it means.

There is token *[UNK]* that stands for unknown token in paraphrase. Ones that contains *[UNK]* should have both grammar and coherent score lower than 5. The grammar score should depend on other tokens in the paraphrase. The specific coherent score depends on the impact of *[UNK]* on that certain paraphrase. Here are some paraphrase examples given original question *how can robot have human intelligence ?*:

- *paraphrase:* how can [UNK] be intelligent ?
  coherent score: 1
  This token prevent us from understanding the question and give proper answer. It causes serious information loss here;

- *paraphrase:* how can robot [UNK] intelligent ?
  coherent score: 3
  There is information loss, but the unknown token does not influence our understanding so much;

- *paraphrase:* how can robot be intelligent [UNK] ?
  coherent score: 4
  [UNK] basically does not influence understanding.

NOTED:

- Please decouple *grammar* and *coherent* as possible as you can. For instance, given a sentence *is it true that girls like shopping*, the paraphrase *do girls like go go shopping* can get a *coherent* score of 5 but a *grammar* score of only 3. But for the one you even can not understand, e.g., *how is the go shopping of girls*, you should give both of low *grammar* score and low *coherent* score, even it contains some topic-relevant words.

- Do a Google search when you see any strange entity name such that you can make more appropriate judgement.