

Paraphrase Generation with Deep Reinforcement Learning

Zichao Li¹, Xin Jiang¹, Lifeng Shang¹, Hang Li²

¹Noah's Ark Lab, Huawei Technologies

{li.zichao, jiang.xin, shang.lifeng}@huawei.com

²Toutiao 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), and plays a key role in a number of applications such as question answering, search, and dialogue. 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. Empirical study shows that the learned evaluator can guide the generator to produce more accurate paraphrases. Experimental results demonstrate the proposed models (the generators) 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*” and “*how many miles is it from Earth to Sun*” are paraphrases. Paraphrase generation refers to a task in which given a sentence one creates a paraphrase (or paraphrases) of it. Paraphrase generation is an important task in NLP, which can be a key component in many applications such as retrieval-based question answering, query reformulation for web search, data augmentation for task-oriented dialogue. However, due to the complexity 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) and copying mechanism (Vinyals et al.,

2015; Gu et al., 2016; See et al., 2017), 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 approach to paraphrase generation.

The key challenge in paraphrase generation lies in the definition of the evaluation measure. Ideally the measure should be able to decide whether a generated sentence is a paraphrase of the given sentence on the basis of semantic similarity. One could train a sequence-to-sequence model for paraphrase generation using cross entropy as evaluation measure. Cross entropy can only work as a loose approximation of the semantic similarity, however. To tackle this problem, Ranzato et al. (2015) propose using sequence-level measures such as BLEU (Papineni et al., 2002) and ROUGE (Lin, 2004) and employing reinforcement learning (RL), specifically the REINFORCE algorithm (Williams, 1992), to optimize the measures as reward functions. However, the measures are calculated based on the lexical similarity to the human references, which may not perfectly represent semantic similarity for paraphrasing. It is likely that a correctly generated sequence will get a low score due to lexical 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 given the former as the 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 identification). The generator is a sequence-to-sequence (Seq2Seq) learning model, which is first trained by using cross entropy and gradient descent and then fine-tuned by using policy gradient with supervisions from the evaluator as rewards. The evaluator is a deep matching model, which can be trained by using a pointwise loss and gradient descent when both positive and negative examples are available as training data, or by max-margin inverse reinforcement learning with output from the generator as supervisions when only positive examples are available. Our work is inspired by the work of AlphaGo, the computer go system and GAN (generative adversarial networks).

Specifically, for the generator we adopt the attentive Seq2Seq architecture with copy mechanism (Bahdanau et al., 2015; See et al., 2017), and for the evaluator we adopt the deep matching model based on the decomposable attention model (Parikh et al., 2016). In the latter setting, for the training of evaluator using inverse reinforcement learning (IRL), we develop a novel algorithm based on max-margin IRL principle (Ratliff et al., 2006). For the training of generator, we employ a curriculum learning strategy (Bengio et al., 2009) to steadily train the generator to improve its performance. The generator can be further fine-tuned with unparalleled data, which is particularly effective when the amount of labeled data is small.

We evaluate the effectiveness of our approach using the Quora question pair dataset through both automatic and human assessments. We find that the well-trained evaluator 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 in paraphrase generation than 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 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 implementation techniques including the reward rescaling and curriculum learning.

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

Paraphrase generation can be regarded as a sequence-to-sequence transformation problem. Given an input sequence of words $X = [x_1, \dots, x_L]$ with length L , we consider generating another output sequence of words $Y = [y_1, \dots, y_T]$ with length T that has the same meaning as X . There are different levels of granularities from phrase to text, 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.

Although obtaining fairly good results on paraphrasing (e.g., Cao et al. (2017); Prakash et al. (2016); Ranzato et al. (2015); Bahdanau et al. (2016)), existing sequence-to-sequence learning methods, which use the cross-entropy or the lexical similarity measures such as BLEU and ROUGE as the objectives, is still 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 transformation in terms of cross-entropy or BLEU and ROUGE. The evaluation measure in the training and prediction phases should accurately represent semantic similarity between sentences.

To attack the above problem, we propose a new approach to paraphrase generation using deep reinforcement learning. The key idea is to employ a sequence-to-sequence learning model for paraphrase generation, and train the model using first deep learning and then reinforcement learning guided by a deep matching model which itself is trained by deep learning or inverse reinforcement learning. The two models are referred to as *generator* and *evaluator*, respectively. 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 approach.

2.2 Generator: Sequence-to-Sequence Learning Model

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

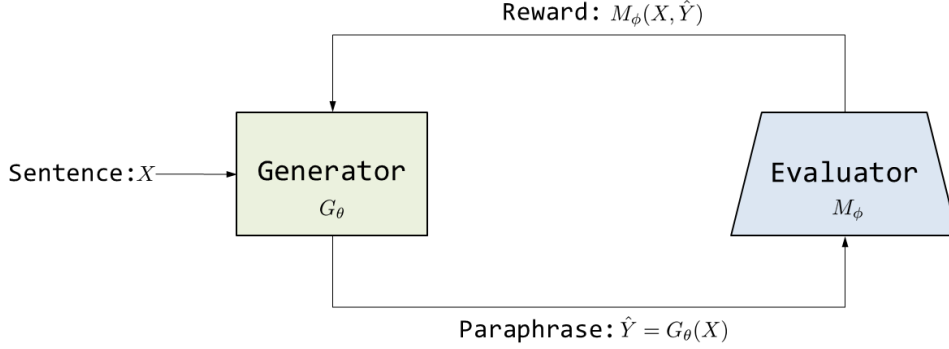


Figure 1: Framework of Deep Reinforcement Learning for Paraphrase Generation.

X , the encoder RNN transforms it to a sequence of hidden states $H = [h_1, \dots, h_L]$, 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:

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

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.

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:

$$p_\theta(y_t | Y_{1:t-1}, X) = q(s_t, c_t, y_{t-1}) g(s_t, c_t, y_{t-1}) + (1 - q(s_t, c_t, y_{t-1})) \sum_{i: y_t = x_i} \alpha_{ti}, \quad (2)$$

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.

2.3 Evaluator: Deep Matching Model

Paraphrase identification, which is to decide whether a pair of sentences are paraphrases, is a short text matching problem. 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)).

In our work, we choose a simple yet effective neural network architecture, called the *decomposable-attention* model (Parikh et al., 2016), to compute the similarity between two sentences. Given two input sentences X and Y , the model conducts matching in three steps as follows.

Attend Two sentences are inter-attended by each other:

$$\begin{aligned}\bar{x}_i &= \sum_{j=1}^T \alpha_{ji} e(y_j), \quad \alpha_{ji} = \frac{\exp e(x_i)^T e(y_j)}{\sum_{k=1}^T \exp e(x_i)^T e(y_k)}, \\ \bar{y}_j &= \sum_{i=1}^L \beta_{ji} e(x_i), \quad \beta_{ji} = \frac{\exp e(x_i)^T e(y_j)}{\sum_{k=1}^L \exp e(x_k)^T e(y_j)}.\end{aligned}\tag{3}$$

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 :

$$\mathbf{x}_i = f_c(e(x_i), \bar{x}_i), \quad \mathbf{y}_j = f_c(e(y_j), \bar{y}_j)\tag{4}$$

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

$$z = f_a \left(\sum_{i=1}^L \mathbf{x}_i, \sum_{j=1}^T \mathbf{y}_j \right),\tag{5}$$

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:

$$e'_k(i) = \begin{cases} \sin(i/10000^{k/d}), & \text{if } k \equiv 0 \pmod{2}, \\ \cos(i/10000^{(k-1)/d}), & \text{if } k \equiv 1 \pmod{2}, \end{cases}\tag{6}$$

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

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 is first trained to maximize the cross entropy, i.e., the conditional likelihood of the output sequence given the input sequence:

$$\mathcal{L}_{\text{seq2seq}}(\theta) = \sum_{t=1}^T \log p_\theta(y_t | Y_{1:t-1}, X).\tag{7}$$

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 technique is called *teacher forcing*.

Learning with teacher forcing, the discrepancy between training and prediction can quickly accumulate errors along the generated sequence (Bengio et al., 2015; Ranzato et al., 2015). Therefore, the generator G_θ is further fine-tuned to maximize the expected cumulative reward in reinforcement learning (RL), which is given by the evaluator.

In the RL setting, generation of the next word is defined as an *action*, and the probability of the generation $p_\theta(y_t|Y_{1:t-1}, X)$ 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, and the goal of RL is to find a policy (accordingly the generator) that maximizes the expected cumulative reward:

$$\mathcal{L}_{RL}(\theta) = \mathbb{E}_{p_\theta(Y_{1:T}|X)} \sum_{t=1}^{t=T} r_t(y_t; X, Y_{1:t-1}) \quad (8)$$

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 $Q_t = Q(y_t; X, \hat{Y}_{1:t-1}) = \mathbb{E}_{p_\theta(Y_{t+1:T}|\hat{Y}_{1:t-1}, y_t, X)} R(\hat{Y})$. The value function can be estimated by aggregating the Monte-Carlo simulations at each time step:

$$Q_t = \begin{cases} \frac{1}{N} \sum_{n=1}^N R([\hat{Y}_{1:t-1}, y_t, \hat{Y}_{t+1:T}^n]), & t < T \\ R([\hat{Y}_{1:t-1}, y_t]), & t = T. \end{cases} \quad (9)$$

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

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

$$\nabla_\theta \hat{\mathcal{L}}_{RL}(\theta) = \sum_{t=1}^T \nabla_\theta \log p_\theta(\hat{y}_t|\hat{Y}_{1:t-1}, X) Q(\hat{y}_t; X, \hat{Y}_{1:t-1}) \quad (10)$$

We can apply the gradient ascent algorithms to maximize the expected cumulative reward based on the equation.

The reward function is defined by the evaluator which can measure the semantic similarity between sentences. In particular, when a pair of input sentence X and generated sentence $\hat{Y} = G_\theta(X)$ is given, the evaluator $M_\phi(X, \hat{Y})$ can output how likely this pair being paraphrases.

3.2 Learning of Evaluator

We propose two methods for learning the evaluator in different settings. When there are a large number of positive and negative examples, 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), we take an inverse reinforcement learning (IRL) approach to learn the evaluator, i.e., the reward function.

3.2.1 Supervised Learning

Given a set of positive and negative examples (paraphrase pairs), we train M_ϕ with the pointwise cross-entropy loss:

$$\mathcal{J}_{SL}(\phi) = -\log M_\phi(X, Y) - \log(1 - M_\phi(X, Y^-)), \quad (11)$$

where Y^- represents a sentence that is not a paraphrase of X . An alternative to (11) is the pairwise ranking loss.

With a well-trained evaluator M_ϕ , we can further train the generator G_θ by reinforcement learning with the output from M_ϕ as reward. The objective function and training method remain the same

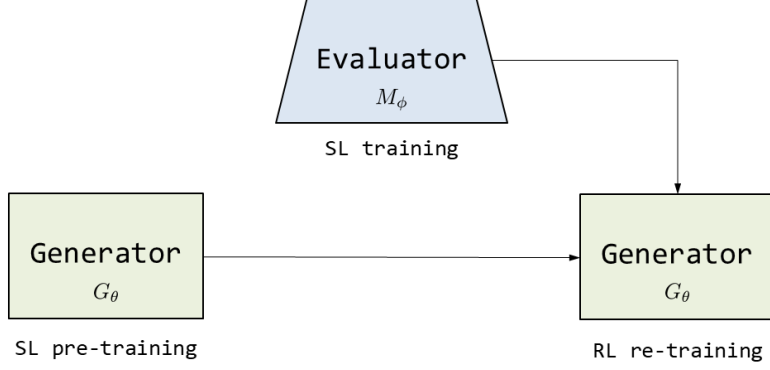


Figure 2: Learning Process of RbM-SL

as in section 3.1. The only difference is to use a different reward, and accordingly a different value function:

$$Q_t = \begin{cases} \frac{1}{N} \sum_{n=1}^N M_\phi(X, [\hat{Y}_{1:t-1}, y_t, \hat{Y}_{t+1:T}^n]), & t < T \\ M_\phi(X, [\hat{Y}_{1:t-1}, y_t]), & t = T. \end{cases} \quad (12)$$

We call this method **RbM-SL** (Reinforced by Matching with Supervised Learning). In the RbM-SL model, the evaluator M_ϕ is expected to capture various paraphrasing patterns and to make precise judgements on whether two sentences are paraphrases. Through maximizing the expected cumulative reward induced from M_ϕ , the generator G_θ can learn to generate more accurate paraphrases. Figure 2 demonstrates learning process of the RbM-SL model. The detailed training procedure is shown in Algorithm 1.

3.2.2 Inverse Reinforcement Learning

The supervised learning method requires the use of positive and negative examples of paraphrase pairs. However, such data might not be always available. With only positive examples, we can still learn the evaluator by employing inverse reinforcement learning (IRL). IRL is the problem of recovering the reward function from the demonstrated behaviors (sequence of states and actions) (Ng et al., 2000; Abbeel & Ng, 2004). IRL has a close connection to generative adversarial networks (GAN), as discussed by Finn et al. (2016a).

In our case the reference paraphrases given by humans are the behaviors. According to the principle of IRL, the goal is to find a reward function R^* under which the human reference Y is nearly optimal among all the paraphrasing policies, i.e.

$$\mathbb{E}_{p_{\theta^*}(Y|X)} R^*(Y) \geq \mathbb{E}_{p_\theta(\hat{Y}|X)} R^*(\hat{Y}), \quad \forall \theta. \quad (13)$$

Once the reward function is found, it is then used to improve the policy $p_\theta(\hat{Y}|X)$ through the standard RL procedure.

In this work, motivated by Ratliff et al. (2006), we formulate the IRL problem as a maximum margin optimization with the objective function:

$$\mathcal{J}_{\text{IRL}}(\phi) = \max(0, 1 + M_\phi(X, \hat{Y}) - M_\phi(X, Y) - \zeta(\hat{Y}, Y)). \quad (14)$$

Here \hat{Y} is the generated sentence by the previous generators. $\zeta(\hat{Y}, Y)$ is the slack variable. In our experiment we set $\zeta(\hat{Y}, Y) = \text{ROUGE-L}(\hat{Y}, Y)$. Minimizing the above objective function induces

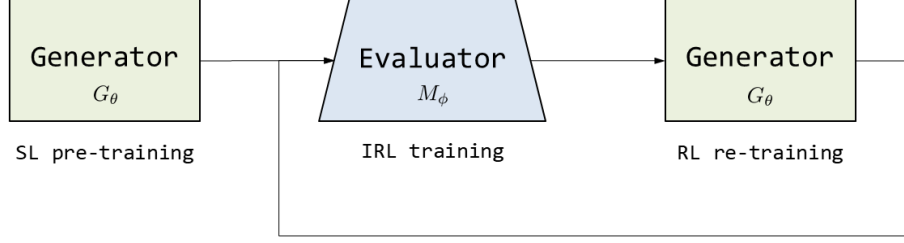


Figure 3: Learning Process of RbM-IRL

the evaluator M_ϕ to assign a higher score to the human reference than to the corresponding generated sentence. We call this method **RbM-IRL** (Reinforced by Matching with Inverse Reinforcement Learning).

To ensure that the generator G_θ generates a paraphrase close to the human reference, the generator G_θ and the evaluator M_ϕ must be trained alternatively. Figure 3 demonstrates learning process of the RbM-IRL model. The detailed training procedure is shown in Algorithm 2. The advantage of RbM-IRL over RbM-SL is obvious. RbM-IRL makes effective use of examples generated during training to learn the evaluator, without the need of additional data. As the generated sentence becomes closer to the ground-truth, the evaluator also tends to be more discriminative in identifying a paraphrase.

It should be noted that in both of RbM-SL and RbM-IRL, once the evaluator is learned, the reinforcement learning of the generator only requires sentences as input. This allows the generator being further trained with unparalleled text, which can potentially enhance its generalization ability. From another point of view, combined with the deep matching model, the Seq2Seq model can make full use of information other than a parallel corpus, such as negative examples, preferences relations, and unsupervised data.

3.3 Training Techniques

3.3.1 Reward Rescaling

In practice reinforcement learning algorithms often suffer from instability in training. A common approach to reduce the gradient variance is to subtract a baseline reward from the value function. For instance, a simple baseline can be a moving average of historical rewards. More advanced methods like the actor-critic algorithms employ another trainable model (the critic) to approximate the expected reward. While in our work, especially in RbM-IRL, the reward function (the evaluator) keeps updating during training. Thus, keeping track of a baseline reward is unstable and inefficient. Inspired by Guo et al. (2017)’s work, we propose an efficient reward rescaling method based on ranking. For a batch of D generated paraphrases $\{\hat{Y}^1, \dots, \hat{Y}^D\}$, each associated with a reward $R^d = M_\phi(X^d, \hat{Y}^d)$, we rescale these rewards by

$$\bar{R}^d = \sigma(\delta_1 \cdot (0.5 - \frac{\text{rank}(d)}{D})) - 0.5, \quad (15)$$

where $\sigma(\cdot)$ is sigmoid function, $\text{rank}(d)$ is the rank of R^d in $\{R^1, \dots, R^D\}$, and δ_1 is the scalar controlling the variance of rewards. Similar strategy is applied in estimating in-sequence value function for each token, and the final rescaled value function is

$$\bar{Q}_t^d = \sigma(\delta_2 \cdot (0.5 - \frac{\text{rank}(t)}{T})) - 0.5 + \bar{R}^d, \quad (16)$$

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 (unparalleled) sentences $\{X\}$.

Output: Generator $G_{\theta'}$

- 1 Train the evaluator M_ϕ with $\{(X, Y)\}$ and $\{(X, Y^-)\}$;
- 2 Pre-train the generator G_θ with $\{(X, Y)\}$;
- 3 Init $G_{\theta'} := G_\theta$;
- 4 **while** *not converge* **do**
- 5 Sample a sentence $X = [x_1, \dots, x_L]$ from the paraphrase corpus or the unparalleled corpus;
- 6 Generate a sentence $\hat{Y} = [\hat{y}_1, \dots, \hat{y}_T]$ according to $G_{\theta'}$ given input X ;
- 7 Set the gradient $g_{\theta'} = 0$;
- 8 **for** $t = 1$ **to** T **do**
- 9 Run N Monte Carlo simulations: $\{\hat{Y}_{t+1:T}^1, \dots, \hat{Y}_{t+1:T}^N\} \sim p_{\theta'}(Y_{t+1:T} | \hat{Y}_{1:t-1}, \hat{y}_t, X)$;
- 10 Compute the value function Q_t by (12);
- 11 Rescale the reward to \bar{Q}_t by (16);
- 12 Accumulate θ' -gradient: $g_{\theta'} := g_{\theta'} + \nabla_\theta \log p_{\theta'}(\hat{y}_t | \hat{Y}_{1:t-1}, X) \bar{Q}_t$
- 13 **end**
- 14 Update $G_{\theta'}$ using the gradient $g_{\theta'}$ with learning rate γ_G : $G_{\theta'} := G_{\theta'} + \gamma_G g_{\theta'}$
- 15 **end**
- 16 **Return** $G_{\theta'}$

Algorithm 2: Training Procedure of RbM-IRL

Input : A corpus of paraphrase pairs $\{(X, Y)\}$, a corpus of (unparalleled) sentences $\{X\}$.

Output: Generator $G_{\theta'}$, evaluator $M_{\phi'}$

- 1 Pre-train the generator G_θ with $\{(X, Y)\}$;
- 2 Init $G_{\theta'} := G_\theta$ and $M_{\phi'}$;
- 3 **while** *not converge* **do**
- 4 **while** *not converge* **do**
- 5 Sample a sentence $X = [x_1, \dots, x_L]$ from the paraphrase corpus;
- 6 Generate a sentence $\hat{Y} = [\hat{y}_1, \dots, \hat{y}_T]$ according to $G_{\theta'}$ given input X ;
- 7 Calculate ϕ' -gradient: $g_{\phi'} := \nabla_\phi \mathcal{J}_{\text{IRL-CL}}(\phi)$;
- 8 Update $M_{\phi'}$ using the gradient $g_{\phi'}$ with learning rate γ_M : $M_{\phi'} := M_{\phi'} - \gamma_M g_{\phi'}$
- 9 **end**
- 10 Train $G_{\theta'}$ with $M_{\phi'}$ as in Algorithm 1;
- 11 **end**
- 12 **Return** $G_{\theta'}, M_{\phi'}$

where $\text{rank}(t)$ is the rank of Q_t^d in $\{Q_1^d, \dots, Q_T^d\}$.

Reward rescaling has two advantages. First, the mean and variance of \bar{Q}_t^d 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 in terms of efficiency and accuracy, but not in RbR. The reason seems to be that the ROUGE score is already an accurate lexicon match evaluation.

3.3.2 Curriculum Learning

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

$$\mathcal{J}_{\text{IRL-CL}}^k(\phi) = w^k \max(0, 1 + M_\phi(X^k, \hat{Y}^k) - M_\phi(X^k, Y^k) - \zeta(\hat{Y}^k, Y^k)) \quad (17)$$

In curriculum learning, w^k is determined by the difficulty of example. At the beginning, the training procedure concentrates on the relatively simple examples, and gradually put more weights on the difficult ones. In our case, we use edit distance between X and Y as the measure of the difficulty. Specifically, w^k is determined by:

$$w^k \sim \text{Binomial}(p^k, 1), \quad p^k = \sigma(\delta_3 \cdot (0.5 - \frac{\text{rank}(\mathcal{E}(X^k, Y^k))}{K})) \quad (18)$$

where $\mathcal{E}(\cdot, \cdot)$ is the edit distance. 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 generator first learns to generate the paraphrases with small modifications on the input sentences (e.g. “*what ’s*” \rightarrow “*what is*”). Along with training it gradually learns more complicated paraphrasing patterns (e.g. “*how can I*” \rightarrow “*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 methods, we choose three baseline models: a vanilla version of (attentive) Seq2Seq model, a Seq2Seq model with copy mechanism, and a Seq2Seq model reinforced by ROUGE-2 (RbR), which is close to that in Ranzato et al. (2015).

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)¹, BLEU-2 (Papineni et al., 2002)² and METEOR (Lavie & Agarwal, 2007)³. For the manual evaluation, we conduct evaluation on the generated paraphrases aggregated from all methods. As pointed out, ideally it would be better not to 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.

¹ <http://www.rxnlp.com/rouge-2-0/>

² <http://www.nltk.org/>

³ <http://www.cs.cmu.edu/~alavie/METEOR/>

4.2 Datasets

We validate our method with the Quora question pair dataset. This is a newly released dataset with more than 400K pairs of questions ⁴. Each question pair is labeled whether the questions are *duplicated* (i.e., they are paraphrases) or not. About 140K of question pairs are labeled as duplicated (positive examples), while the other pairs are not (negative examples). We split the dataset of question pairs in two different ways resulting in two experimental settings: Quora-I and Quora-II. In Quora-I, we randomly partition the set of question pairs into training (100K), testing (30K), and validation (3K) datasets, respectively. In Quora-II, we randomly partition the set of questions pairs into training (50K), testing (30K), and validation (3K) and ensure that there is no shared question between the training and test/validation datasets. Obviously, it is more difficult to achieve higher performance with Quora-II, because the paraphrase questions in the test/validation datasets are unseen in the training datasets.

Positive examples are used in training of the generator for Seq2Seq, RbR, RbM-SL and RbM-IRL. During the RL phrase of RbM-SL and RbM-IRL, questions sampled from the training dataset are also fed to the generator. Both positive and negative examples are used in training of the evaluator for RbM-SL. Only positive examples are used in training of the evaluator for RbM-IRL.

4.3 Implementation Details

4.3.1 Generator

For the generator G_θ , 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 word embeddings with 128 dimensions are trained from scratch. The encoder is built as a bidirectional LSTM (Long-Short-Term-Memory) (Hochreiter & Schmidhuber, 1997) with a unit of 256 dimensions and the decoder is created as an LSTM with a unit of 256 dimensions as well. The generator is trained on a single NVIDIA Tesla K80 GPU using TensorFlow (Abadi et al., 2016), with the batch size of 80. We use Adadgrad optimizer (Duchi et al., 2011) in supervised pre-training and Adam optimizer (Kingma & Ba, 2014) in reinforcement learning. We also fine-tune the 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 $N = 4$. To make the training more stable, we use the ground-truth with reward of 0.1.

4.3.2 Evaluator

For the evaluator M_ϕ in RbM-SL, the word embeddings of M_ϕ are pre-trained by word2vec and fine-tuned afterwards. Specifically, we use the GoogleNews 300-dimension word vectors ⁵, and map them to 200 dimensions as in Parikh et al. (2016). The evaluator M_ϕ is also trained on a single NVIDIA Tesla K80 GPU using TensorFlow, with 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 embeddings as proposed in Vaswani et al. (2017). For the evaluator of M_ϕ in RbM-IRL, the learning rate decreases to $1e-2$. We use the technique of reward rescaling as mentioned in section 3.3.1 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

⁴ <https://www.kaggle.com/c/quora-question-pairs>

⁵ <https://code.google.com/archive/p/word2vec/>

from 12 to 3 and δ_3 from 15 to 8 during curriculum learning. In RbR, we take the exponential moving average of historical rewards as baseline:

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

where b_m is the baseline b at iteration m , and we set λ as 0.1 by grid search.

4.4 Results

Table 1 and table 2 show the performances of the models under different beam-search strategies in Quora-I. Table 3 and table 4 show the performances of the models in Quora-II. 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 paraphrasing. From all automatic measures, RbM-SL is constantly better than RbM-IRL, 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 seems to become more capable in paraphrase identification.

Table 1: Performances of different models (beam size=1) on Quora-I.

Models	ROUGE-1	ROUGE-2	BLEU-2	METEOR
Vanilla Seq2Seq	58.50	30.70	35.91	25.16
Seq2Seq with copy	61.40	35.12	39.90	29.63
Seq2Seq RbR	62.97	36.60	41.39	29.64
Seq2Seq RbM-SL	63.69	37.61	42.85	31.78
Seq2Seq RbM-IRL	63.63	37.33	42.68	31.62

Table 2: Performances of different models (beam size=10) on Quora-I.

Models	ROUGE-1	ROUGE-2	BLEU-2	METEOR
Vanilla Seq2Seq	58.77	31.47	36.55	26.28
Seq2Seq with Copy	61.96	36.07	40.55	30.21
Seq2Seq RbR	63.35	37.33	41.83	30.96
Seq2Seq RbM-SL	64.39	38.11	43.54	32.84
Seq2Seq RbM-IRL	64.02	37.72	43.09	31.97

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 aggregated and partitioned into seven random buckets for seven human assessors to evaluate. The assessors are asked to rate each sentences 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 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 5 and

Table 3: Performances of different models (beam size=1) on Quora-II.

Models	ROUGE-1	ROUGE-2	BLEU-2	METEOR
Vanilla Seq2Seq	45.79	19.77	24.93	19.89
Seq2Seq with Copy	49.40	22.91	27.62	23.23
Seq2Seq RbR	52.50	25.83	30.70	24.87
Seq2Seq RbM-SL	55.46	29.57	34.09	27.60
Seq2Seq RbM-IRL	55.38	28.52	33.44	26.07

Table 4: Performances of different models (beam size=10) on Quora-II.

Models	ROUGE-1	ROUGE-2	BLEU-2	METEOR
Vanilla Seq2Seq	47.22	20.72	26.06	20.35
Seq2Seq with Copy	51.98	25.16	30.01	24.31
Seq2Seq RbR	54.50	27.50	32.54	25.67
Seq2Seq RbM-SL	57.34	31.09	35.81	28.12
Seq2Seq RbM-IRL	56.86	29.90	34.79	26.67

table 6 demonstrate 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. This may indicate that RbM-IRL can learn the language modeling capability better.

Table 5: Human evaluation results on Quora-I

Models	Relevance	Fluency
Seq2Seq with Copy	3.23	4.55
Seq2Seq RbR	3.56	4.61
Seq2Seq RbM-SL	4.08	4.67
Seq2Seq RbM-IRL	4.07	4.69
Ground-truth Reference	4.69	4.95

Table 6: Human evaluation results on Quora-II

Models	Relevance	Fluency
Seq2Seq with Copy	2.34	2.96
Seq2Seq RbR	2.58	3.14
Seq2Seq RbM-SL	3.20	3.48
Seq2Seq RbM-IRL	2.80	3.53
Ground-truth Reference	4.68	4.90

Figure 4 gives some examples of top generated paraphrases by the RbM-SL and RbM-IRL models on the Quora-I.

Figure 4: Examples of the generated paraphrases by RbM-SL model on Quora-I.

Models	Questions	Top Generated Paraphrases
RbM-SL	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 ?
RbM-SL	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 ?
RbM-IRL	what will be the impact of brexit on the us economy ?	how can the impact of brexit be on the us economy ?
		how will the us brexit affect the us economy ?
		how will brexit affect the us economy ?
RbM-IRL	how do i find my social security number online for free ?	what is the best way to find a social security number online ?
		what are some of the best strategies to find your social security number online ?
		what is the best way to get my social security number ?

5 Related Work

5.1 Sequence-to-Sequence Learning

Sequence-to-sequence learning (or the encoder-decoder framework) is first proposed by Sutskever et al. (2014); Cho et al. (2014), denoted as Seq2Seq. It learns a model that can map one sequence of words to another sequence of words. It has been widely applied into 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)). The attention mechanism is introduced into Seq2Seq to solve the alignment problem in machine translation and dialogue generation Bahdanau et al. (2015). Furthermore, the copy mechanism is added to Seq2Seq to tackle the out of vocabulary problem in document summarization (Gu et al., 2016; See et al., 2017). The insertion mechanism is included into Seq2Seq to incorporate an expression of fact into the sentence in generative question answering (Yin et al., 2015). Most of the Seq2Seq models are learned by using the cross entropy loss and gradient descent. In Ranzato et al. (2015); Bahdanau et al. (2016), the learning of Seq2Seq model is formalized as a reinforcement learning problem, in which the reward function is manually defined and policy gradient is utilized. In our work, we define the generator for paraphrase generation as a Seq2Seq model with attention and copy mechanism. We basically adopt the model proposed in See et al. (2017) for its simplicity. The generator is first trained by gradient descent and then by policy gradient. Our work is unique in that the reward function is also trained by another neural network, i.e., the evaluator.

5.2 Paraphrase Generation

Paraphrase generation has been intensively studied due to its importance. There are several works on paraphrase generation using deep learning, particularly, sequence-to-sequence learning, in addition to the traditional approaches. For example, 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 generation. Gupta et al. (2017) use variational auto-encoder framework to generate more diverse paraphrases. Zhang & Lapata (2017) tackle a similar task of sentence simplification with Seq2Seq model coupled with deep reinforcement learning, in which reward function is manually defined for the task. Similar to these works, we pre-train the generator within the Seq2Seq framework. However, there is also striking difference between our work and existing works. we employ two models, a generator for paraphrase generation and an evaluator for paraphrase identification, as well as deep reinforcement learning to train the models.

5.3 Deep Matching Models

Deep neural networks for matching, referred to as deep matching models in this paper, have been developed for ‘semantic matching’ between two short texts. They are essential for a variety of applications such as question answering and search. The models can also be applied to paraphrase identification, including the RNN-based model (Wang & Jiang, 2015), CNN-based model (Hu et al., 2014) and attention-based model (Parikh et al., 2016). We use the attention-based model (Parikh et al., 2016) in this work, because it is quite efficient and effective in paraphrase identification. Inspired by the technique in Vaswani et al. (2017), we also incorporate word order information into the model by position embedding.

5.4 Inverse Reinforcement Learning

Inverse reinforcement learning (IRL) is a sub-field in reinforcement learning (RL). The goal of IRL is to learn an optimal reward function given a sequence of states and actions from an *expert policy*. The basic principle of IRL is to estimate a reward function under which the expert policy outperforms the other policies. There are different approaches to IRL. One approach is *feature expectation matching* (FEM), proposed in Abbeel & Ng (2004) and further studied in Ziebart et al. (2008); Finn et al. (2016b); Ho et al. (2016). FEM assumes the reward function expressed as a linear combination of known features, and is solved by min-max optimization. Our work is based on another approach, referred to as maximum-margin IRL (Ratliff et al., 2006), which manages to maximize the reward difference between the expert and other policies in learning. There does not seem to be much work on IRL for NLP. In Neu & Szepesvári (2009), parsing is formalized as an FEM problem. To our knowledge, our work is the first that applies IRL into the sequence-to-sequence learning tasks.

5.5 Alpha Go System

AlphaGo (Silver et al., 2016) is the well-known computer go system which has defeated human world champions. AlphaGo uses a *policy network* to decide the next action in given a state and a *value network* to evaluate the value of an action given a state. In the game, the policy network and value network are combined with Monte Carlo tree search to select the best move to take at each step, which is most likely to lead to a win. The architecture of AlphaGo has inspired us to design the generator-evaluator framework in our work for sequence-to-sequence learning.

5.6 Generative Adversarial Networks

Recently a new framework for data generation called Generative Adversarial Net (GAN) (Goodfellow et al., 2014) is gaining popularity. GAN contains a discriminator and a generator. The generator tries to map the distribution of noisy data to the true distribution of data of interest, while the discriminator tries to decide whether a sample is generated from the generator or the true distribution. A more

robust generator can be learned through the adversarial learning process of GAN. Recently the relation between GAN and IRL is pointed out by Finn et al. (2016a). There are applications of GAN to NLP, such as text generation (Yu et al., 2017; Guo et al., 2017) and dialogue generation (Li et al., 2017). Our framework for paraphrase generation also contains two models, and the generator creates a paraphrase given a sentence. Therefore, there is similarity between our framework and GAN.

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 (the generator) 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.

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.
- Pieter Abbeel and Andrew Y Ng. Apprenticeship learning via inverse reinforcement learning. In *Proceedings of the twenty-first international conference on Machine learning*, pp. 1. ACM, 2004.
- 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.
- Yoshua Bengio, Jérôme Louradour, Ronan Collobert, and Jason Weston. Curriculum learning. In *Proceedings of the 26th annual international conference on machine learning*, pp. 41–48. ACM, 2009.

- Igor Bolshakov and Alexander Gelbukh. Synonymous paraphrasing using wordnet and internet. *Natural Language Processing and Information Systems*, pp. 189–200, 2004.
- 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.
- Chelsea Finn, Paul Christiano, Pieter Abbeel, and Sergey Levine. A connection between generative adversarial networks, inverse reinforcement learning, and energy-based models. *arXiv preprint arXiv:1611.03852*, 2016a.
- Chelsea Finn, Sergey Levine, and Pieter Abbeel. Guided cost learning: Deep inverse optimal control via policy optimization. In *International Conference on Machine Learning*, pp. 49–58, 2016b.
- 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.
- Jiaxian Guo, Sidi Lu, Han Cai, Weinan Zhang, Yong Yu, and Jun Wang. Long text generation via adversarial training with leaked information. *arXiv preprint arXiv:1709.08624*, 2017.
- 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.
- Jonathan Ho, Jayesh Gupta, and Stefano Ermon. Model-free imitation learning with policy optimization. In *International Conference on Machine Learning*, pp. 2760–2769, 2016.
- 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.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.
- 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.
- Gergely Neu and Csaba Szepesvári. Training parsers by inverse reinforcement learning. *Machine learning*, 77(2):303–337, 2009.
- 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.
- Andrew Y Ng, Stuart J Russell, et al. Algorithms for inverse reinforcement learning. In *ICML*, pp. 663–670, 2000.
- 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. In *COLING*, 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.

- Nathan D Ratliff, J Andrew Bagnell, and Martin A Zinkevich. Maximum margin planning. In *Proceedings of the 23rd international conference on Machine learning*, pp. 729–736. ACM, 2006.
- 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.
- David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016.
- 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.
- 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.
- Xingxing Zhang and Mirella Lapata. Sentence simplification with deep reinforcement learning. In *EMNLP*, 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.
- Brian D Ziebart, Andrew L Maas, J Andrew Bagnell, and Anind K Dey. Maximum entropy inverse reinforcement learning. In *AAAI*, volume 8, pp. 1433–1438. Chicago, IL, USA, 2008.