Unsupervised Paraphrasing via Deep Reinforcement Learning

A. B. Siddique
University of California, Riverside
msidd005@ucr.edu

Samet Oymak
University of California, Riverside
oymak@ece.ucr.edu

Vagelis Hristidis
University of California, Riverside
vagelis@cs.ucr.edu

ABSTRACT

Paraphrasing is expressing the meaning of an input sentence in different wording while maintaining fluency (i.e., grammatical and syntactical correctness). Most existing work on paraphrasing use supervised models that are limited to specific domains (e.g., image captions). Such models can neither be straightforwardly transferred to other domains nor generalize well, and creating labeled training data for new domains is expensive and laborious. The need for paraphrasing across different domains and the scarcity of labeled training data in many such domains call for exploring unsupervised paraphrase generation methods. We propose Progressive Unsupervised Paraphrasing (PUP): a novel unsupervised paraphrase generation method based on deep reinforcement learning (DRL). PUP uses a variational autoencoder (trained using a non-parallel corpus) to generate a seed paraphrase that warm-starts the DRL model. Then, PUP progressively tunes the seed paraphrase guided by our novel reward function which combines semantic adequacy, language fluency, and expression diversity measures to quantify the quality of the generated paraphrases in each iteration without needing parallel sentences. Our extensive experimental evaluation shows that PUP outperforms unsupervised state-of-the-art paraphrasing techniques in terms of both automatic metrics and user studies on four real datasets. We also show that PUP outperforms domain-adapted supervised algorithms on several datasets. Our evaluation also shows that PUP achieves a great trade-off between semantic similarity and diversity of expression.

CCS CONCEPTS

• **Computing methodologies** → **Natural language generation**; **Unsupervised learning**; **Reinforcement learning**; **Natural language processing**; *Discrete space search.*

KEYWORDS

Unsupervised paraphrasing; deep reinforcement learning; natural language generation; natural language processing.

ACM Reference Format:
A. B. Siddique, Samet Oymak, and Vagelis Hristidis. 2020. Unsupervised Paraphrasing via Deep Reinforcement Learning. In *Proceedings of the 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining USB Stick (KDD '20), August 23–27, 2020, Virtual Event, USA.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3394486.3403231

KDD '20, August 23–27, 2020, Virtual Event, USA

ACM ISBN 978-1-4503-7998-4/20/08.
https://doi.org/10.1145/3394486.3403231

1 INTRODUCTION

Paraphrasing is the task of generating a fluent output sentence, given an input sentence, to convey the same meaning in different wording. It is an important problem in Natural Language Processing (NLP) with a wide range of applications such as summarization [20], information retrieval [21], question answering [28], and conversational agents [38]. Most of the previous paraphrasing work [15, 23, 34] has focused on *supervised* paraphrasing methods, which require large corpora of parallel sentences (i.e., input and corresponding paraphrased sentences) for training. Unlike large datasets in neural machine translation, there are not many parallel corpora for paraphrasing, and they are often domain-specific, e.g., Quora is a questions dataset, and MSCOCO is an image captioning dataset. Acquiring big parallel datasets for paraphrasing across many domains is not scalable because it is expensive and laborious. Moreover, a model trained in one domain does not generalize well to other domains [24].

The abundance of domains and applications that could benefit from paraphrasing calls for exploring unsupervised paraphrasing, which is still in its infancy. There are relatively few works on unsupervised paraphrasing such as Variational Autoencoder (VAE) [5], Constrained Sentence Generation by Metropolis-Hastings Sampling (CGMH) [30], and Unsupervised Paraphrasing by Simulated Annealing (UPSA) [26]. Although unsupervised approaches have shown promising results, the probabilistic sampling based approaches such as VAE [5] and CGMH [30] are less constrained, and they produce paraphrases that lack semantic similarity to the input. On the other hand, UPSA [26] does not effectively explore the entire sentence space, resulting in paraphrases that are not different enough from the input.

Given the success of Deep Reinforcement Learning (DRL) [43] in a wide range of applications such as Atari games [31], alphaZero [39], and supervised paraphrasing [23], can DRL also help boost the performance of unsupervised paraphrase generation? To the best of our knowledge, this is the first work to employ DRL in unsupervised paraphrase generation, which is challenging due to the following reasons: (*i*) DRL is known to not work well with large vocabulary sizes when starting with a random policy (i.e., random exploration strategy) [10, 23]; (*ii*) paraphrasing is a multi-step (word-by-word) prediction task, where a small error at an early time-step may lead to poor predictions for the rest of the sentence, as the error is compounded over the next token predictions; and (*iii*) it is challenging to define a reward function that incorporates all the characteristics of a good paraphrase with no access to parallel sentences (i.e., the unsupervised setting).

Our proposed method, *Progressive Unsupervised Paraphrasing (PUP)*, progressively trains a DRL-based model for unsupervised paraphrasing and addresses the aforementioned three challenges using the following techniques:

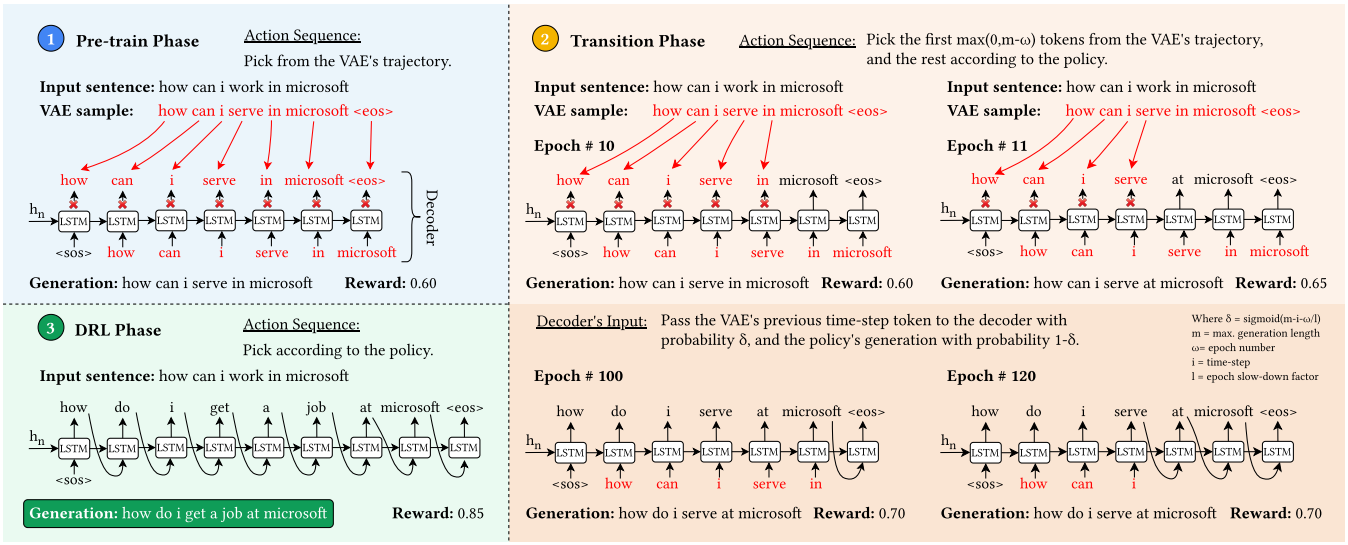

Figure 1: Illustration of the decoding process of the proposed unsupervised paraphrasing method: PUP. Red and black color tokens represent the output from VAE and the DRL's chosen action sequences respectively. Whereas the sentence in green is the final paraphrased sentence generated by PUP for the given input sentence.

• **Unsupervised warm-start of DRL:** PUP warm-starts reinforcement learning by an unsupervised pre-trained VAE [5], which acts as an expert [9, 36] in the pre-training phase. The pre-trained VAE saves the DRL model from expensive global exploration during the initial training phase. Remarkably, the proposed technique is the first instance that can successfully warm-start DRL with an unsupervised model. At the end of DRL training, our DRL model achieves up to 54% higher reward compared to the initial VAE model. We expect that our idea of warm-starting DRL models in an unsupervised fashion may have implications on a broader range of NLP problems with limited labels.

• **Progressive transition for seq2seq DRL:** Another major issue DRL models face is the accumulation of error over the predictions of future tokens. This is particularly significant during the initial exploration of the space. To overcome this, we use a progressive transition that takes advantage of the Sequence-to-Sequence (seq2seq) [42] nature of the problem by transitioning between algorithms (e.g., VAE to DRL) token by token, as shown in Figure 1. Instead of taking actions according to the initial policy (i.e., random action), the model chooses VAE's output as the action, and then incrementally (i.e., one token per epoch) allows the agent to take actions according to the DRL policy. This technique greatly facilitates the convergence of DRL to models with high rewards and is at the heart of the success of DRL.

• **Unsupervised reward function for paraphrasing:** We propose a novel reward function for the DRL model that can measure the quality of the generated paraphrases when no parallel sentences are available. This is accomplished by incorporating the most desirable qualities of a good paraphrase , informed on the paraphrasing literature [8, 29, 41, 48–50]. Our reward function is a combination of semantic adequacy, language fluency, and diversity in expression.

Figure 1 provides an illustration of the decoding process of PUP. First, the decoder of the DRL model relies on the VAE's sample to pick its actions in the pre-train phase. Then, in the transition phase, the model gradually starts taking actions according to its policy. Finally, in the DRL phase, the model picks actions entirely according to its policy to maximize the expected reward. For example, when our DRL model is pre-trained with the VAE sample *"how can i serve in microsoft"*, our fully-trained DRL model amazingly generates the paraphrase *"how do i get a job at microsoft"*.

We evaluate PUP on four real datasets and compare it against state-of-the-art unsupervised paraphrasing techniques; we show that PUP outperforms them in all standard metrics. We also conduct a human study, which demonstrates that human evaluators find PUP's paraphrases to be of higher quality compared to other methods' paraphrases across several carefully selected measures. Moreover, we consider comparisons against domain-adapted models – i.e., models trained on one dataset such as Quora in a supervised setting and then domain-adapted for another dataset such WikiAnswers in an unsupervised fashion. Remarkably, PUP outperforms domain-adapted supervised paraphrasing methods in datasets where applicable.

The rest of the paper is organized as follows. Background is discussed in Section 2, and an overview of PUP is presented in Section 3. The details of PUP are described in Section 4. Sections 5 and 6 present the experimental setup and results, respectively. Section 7 presents the related work, and Section 8 concludes the paper.

2 BACKGROUND

2.1 Encoder-Decoder Framework

An encoder-decoder model (e.g., seq2seq) strives to generate a target sequence (i.e., paraphrase) $Y = (y_1, y_2, \cdots, y_m)$ given an input sequence $X = (x_1, x_2, \cdots, x_n)$, where m and n are target and input sequence lengths respectively. First, the encoder transforms the input sequence X into a sequence of hidden states $(h_1, h_2, \cdots, h_n)$ employing RNN units such as Long Short-Term Memory (LSTM) [17].

The encoder reads the input sequence, one token at a time, until the end of the input sequence token occurs and converts it to hidden state $h_i = Encoder(h_{i-1}, emb(x_i))$ by considering the word embedding of the input token x_i and the previous hidden state h_{i-1} at time-step i. $Encoder(.)$ is a non-linear mapping function and $emb(.)$ maps the given word into a high dimensional space. The decoder utilizes another RNN to generate the paraphrased (i.e., target) sequence Y. The decoder is initialized with the last hidden state h_n, and generates one token at a time, until the end of sentence token (i.e., $< eos >$) is generated. At time-step i, the generation is conditioned on the previously generated words $\hat{y}_{i-1}, \cdots, \hat{y}_1$ and the current decoder hidden state h'_i:

$$P(y_i|\hat{y}_{i-1}, \cdots, \hat{y}_1, X) = softmax(Decoder(h'_i, y_{i-1})). \quad (1)$$

Where $Decoder(.)$ is a non-linear mapping function and $softmax(.)$ converts the given vector into a probability distribution. Such an encoder-decoder model is typically trained by minimizing the negative log-likelihood of the input-target pairs. However, since we do not have access to target sentences in the unsupervised paraphrase generation task, we utilize the reinforcement learning framework.

2.2 VAE: Variational Autoencoder

VAE [19, 35] is a deep generative model for learning a nonlinear latent representation z from data points X. It is trained in an unsupervised fashion for the following loss function:

$$\mathcal{L}_{vae}(\varphi, \phi) = -\mathbb{E}_{q_\varphi(z|X)}\big[\log p_\phi(X|z)\big] + \mathbb{KL}(q_\varphi(z|X)||p(z)), \quad (2)$$

where $q_\varphi(z|X)$ is the encoder with parameters φ that encodes the data points X into a stochastic latent representation z; $p_\phi(X|z)$ is the decoder with the parameters ϕ that strives to generate an observation X given the random latent code z; and $p(z)$ is prior distribution, i.e., standard normal distribution $\mathcal{N}(0, \mathrm{I})$. The first term in Equation 2 is the negative log-likelihood loss for the reconstruction of the data points X. The second term is used to measure Kullback-Leibler ($\mathbb{KL}$) divergence between the the encoder's distribution $q_\varphi(z|X)$ and the prior distribution $p(z)$. At inference time, sentences are sampled [5] from the learned latent representation z. In this work, VAE is employed to provide a warm-start to the DRL-based paraphrasing model so that it does not start from a random policy.

3 OVERVIEW OF PUP

This section provides an overview of the progressive training phases of PUP (Figure 1). It consists of three phases: pre-train, progressive transition, and DRL.

Pre-train phase: For tasks like unsupervised paraphrasing, the big vocabulary impedes the learning process of DRL models. It becomes practically infeasible to train such a model based on the reward alone. To address this issue, we employ a pre-trained VAE (trained on a non-parallel corpus) to provide a warm-start to the DRL model. That is, the output of VAE is used to pick action sequences instead of the agent policy's output. We can think of it as demonstrating the expert's (VAE) actions to DRL, where the expert is an unsupervised model.

Progressive transition phase: The next critical step is to gracefully transition from following the expert's actions to taking actions

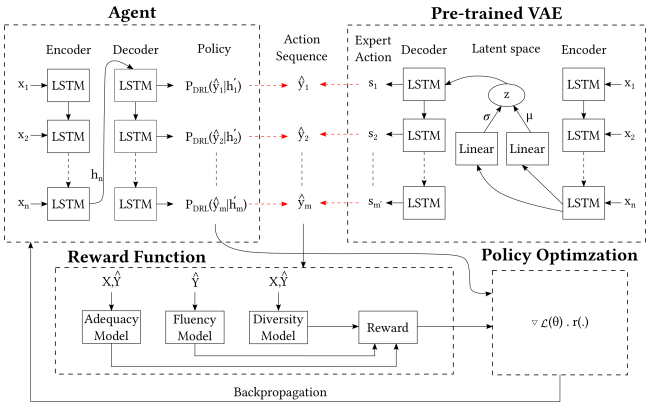

Figure 2: Deep reinforcement learning paradigm for unsupervised paraphrase generation.

according to the policy (i.e., DRL decoder's distribution). An abrupt transition can obstruct the learning process due to the nature of the task, i.e., multi-step prediction, where error accumulates. Especially, an inappropriate sample at an early stage of the sentence (i.e., first few words) may lead to a poor eventual paraphrase generation (i.e., ungrammatical or semantically unfaithful). We propose an intuitive way to pick the first $max(0, m - \omega)$ tokens from VAE's output, and pick the rest according to the agent policy, where m is the length of the generated sentence and ω is the epoch number. Moreover, we pass the output of VAE to the decoder's next time-step with a decreasing probability δ (i.e., decreasing with respect to ω), and the DRL's generation otherwise. This helps with mitigating the accumulation of error, especially in the beginning of the transition phase when the model is expected to make mistakes.

DRL phase: Finally, the model is trained to produce an optimized policy by sampling sentences according to its policy and maximizing its expected reward, which is a combination of the semantic adequacy, language fluency, and diversity in expression.

Figure 2 presents an overview of the DRL paradigm, where action sequences are picked either from VAE's output or the agent policy (highlighted by red dashed arrows) depending on the different phases.

4 PROGRESSIVE UNSUPERVISED PARAPHRASING (PUP)

We first describe how to incorporate DRL for the unsupervised paraphrasing task, then the proposed reward function, and finally we describe the details of PUP.

4.1 Reinforcement Learning Paradigm

The reinforcement learning paradigm for unsupervised paraphrasing is presented in Figure 2. In DRL terminology, the encoder-decoder model (Section 2.1) acts as an agent, which first encodes the input sentence X and then generates the paraphrased version $\hat{Y}$. At time-step i, the agent takes an action $\hat{y}_i \in V$ according to the policy $P_{DRL}(\hat{y}_i|\hat{y}_{1:i-1}, X)$ (see Equation 1), where V represents the possible action space (i.e., vocabulary for generation). The hidden states of the encoder and the previous outputs of the decoder constitute the state. The agent (i.e., model) keeps generating one

token at a time, until the end of sentence token (i.e., < eos >) is produced, which completes the action sequence (i.e., trajectory) $\hat{Y} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$. The policy is optimized by maximizing the expected reward r for the action sequences.

4.2 Paraphrasing Reward

Automatic quality measures for machine translation (or paraphrasing) such as BLEU [32], Rouge [18], TER [40], and METEOR [2] only work when parallel sentences (i.e., targets or references) are available. We propose a novel reward function that incorporates all the characteristics of a good paraphrase and does not require parallel sentences. The most desired qualities of a good paraphrase [8, 29, 41, 48–50] include: semantic adequacy (i.e., similarity in meaning), language fluency (i.e., grammatical correctness), and diversity of expression (i.e., sentence dissimilarity). We define the reward $r(X, \hat{Y})$ of an output sequence $\hat{Y}$ generated by the DRL model for input X as a combination of the above components:

$$r(X, \hat{Y}) = \alpha . r_{Sim}(X, \hat{Y}) + \beta . r_F(\hat{Y}) + \gamma . r_D(X, \hat{Y}), \tag{3}$$

where $r_{Sim}(X, \hat{Y})$, $r_F(\hat{Y})$ and, $r_D(X, \hat{Y}) \in [0, 1]$. $r_{Sim}(X, \hat{Y})$ is the semantic similarity between input X and generated paraphrase $\hat{Y}$. $r_F(\hat{Y})$ captures whether the generated sentence $\hat{Y}$ is grammatically correct or not. $r_D(X, \hat{Y})$ measures the diversity between X and $\hat{Y}$. α, β, and $\gamma \in [0, 1]$ are respective weights. Each component is described below.

Semantic Adequacy: The semantic adequacy reward $r_{Sim}(X, \hat{Y})$ makes sure that the generated paraphrase $\hat{Y}$ is similar in meaning to the input sequence X. We use the universal sentence encoder [7], as it has achieved state-of-art results for semantic textual similarity on the STS Benchmark [6] and it provides a straightforward process to incorporate it in any implementation. In a nutshell, it is trained with a deep averaging network (DAN) encoder, and it generates 512-dimension embedding vector for arbitrary length sentence(s). Then, the semantic similarity can be calculated using the cosine similarity of the vectors v_X and $v_{\hat{Y}}$, which are embedding vectors for the input sequence X and the paraphrased sequence $\hat{Y}$, respectively.

$$r_{Sim}(X, \hat{Y}) = \cos(v_X, v_{\hat{Y}}) = \frac{v_X . v_{\hat{Y}}}{\|v_X\| \|v_{\hat{Y}}\|} \tag{4}$$

Language Fluency: The fluency reward $r_F(\hat{Y})$ measures the grammatical correctness of the generated paraphrase $\hat{Y}$. Since language models such as n-grams [16] and neural models [4] are trained to predict the next token given previous tokens, they can be used to score sentences for fluency. Recently, the Corpus of Linguistic Acceptability (CoLA) [45] has produced the state-of-art results on the grammatical acceptability for in-domain as well as out-of-domain test sets. In its simplest form, CoLA [45] utilizes ELMo-Style (Embeddings from Language Models) and pooling classifier, and it is trained in a supervised fashion. We use a pre-trained CoLA [45] to score our generated paraphrased sequences $\hat{Y}$.

Expression Diversity: The expression diversity reward $r_D(X, \hat{Y})$ encourages the model to generate tokens that are not in the input sequence X. One of the simplest methods to measure the diversity, inverse Jaccard similarity (i.e., $1 - $Jaccard Similarity), could be used. In this work, we use n-grams dissimilarity. To measure the diversity in expression, we use the inverse BLEU of input sequence X and the generated sequence $\hat{Y}$, which is computed using 1 - BLEU(X, $\hat{Y}$). The average of the uni-gram and bi-gram inverse BLEU scores are used in $r_D(X, \hat{Y})$.

Combining the three components: In practice, a reward function that can force the DRL model to generate good quality paraphrases must maintain a good balance across the reward components (i.e., semantic similarity, fluency, and diversity). For example, generating diverse words at the expense of losing too much on the semantic adequacy or fluency is not desirable. Similarly, copying the input sentence as-is to the generation is clearly not a paraphrase (i.e., cosine similarity = 1). To achieve this, we impose strict criteria on the components of the reward function as given below:

$$r_{Sim}(X, \hat{Y}) = \begin{cases} r_{Sim}(X, \hat{Y}), & \text{if } \tau_{min} \leq r_{Sim}(X, \hat{Y}) \leq \tau_{max} \\ 0, & \text{otherwise} \end{cases} \tag{5}$$

$$r_F(\hat{Y}) = \begin{cases} r_F(\hat{Y}), & \text{if } r_F(\hat{Y}) \geq \lambda_{min} \\ 0, & \text{otherwise} \end{cases} \tag{6}$$

$$r_D(X, \hat{Y}) = \begin{cases} r_D(X, \hat{Y}), & \text{if } r_{Sim}(X, \hat{Y}) \geq \tau_{min}, r_F(\hat{Y}) \geq \lambda_{min} \\ 0, & \text{otherwise} \end{cases} \tag{7}$$

Equation 5 makes sure that the model does not copy the input sentence as-is to the generation (i.e., condition: $r_{Sim}(X, \hat{Y}) \leq \tau_{max}$) to enforce the diversity in expression, and does not generate random sentence, which has very low similarity with the input (i.e., condition: $r_{Sim}(X, \hat{Y}) > \tau_{min}$). Equation 6 penalizes the generations that are not fluent. Finally, diverse words (i.e., Equation 7) get rewarded only if the generated sentence achieves a reasonable score on the semantic similarity (i.e., condition: $r_{Sim}(X, \hat{Y}) \geq \tau_{min}$) and fluency (i.e., condition: $r_F(\hat{Y}) \geq \lambda_{min}$). Note that a diversely expressed output sentence, which is not fluent or is not close in meaning to the input sentence needs penalization so that the model may learn a policy that generates not only diverse sentences but also fluent and semantically similar to the input. The objective of combining all the constraints is to ensure competitive outputs in all metrics and to penalize the model for poor generations on any metric. The weights for each component in the reward (i.e., α, β, and γ), and thresholds (i.e., τ_{min}, τ_{max}, and λ_{min}) for Equations 5, 6, and 7 can be defined based the application needs.

4.3 Progressively Training the DRL

The training algorithm optimizes the policy (i.e., encoder-decoder model's distribution $P_{DRL}(.|X)$) to maximize the expected reward $r(.)$ for the generated action sequence $\hat{Y} = (\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$. The loss for a single sample from the possible action sequences is:

$$\mathcal{L}(\theta) = -\mathbb{E}_{(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)} \sim P_{DRL}(.|X)[r(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)]. \tag{8}$$

The loss is the negative expected reward for the action sequences. Infinite number of possible samples make the expectation calculations infeasible, thus it is approximated [46]. The gradient for the $\mathcal{L}(\theta)$ is:

$$\nabla \mathcal{L}(\theta) \approx \sum_{i=1}^{m} \nabla \log P_{DRL}(\hat{y}_i | \hat{y}_{1:i-1}, X)[r(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)]. \tag{9}$$

The training process for the DRL-based unsupervised paraphrase generation model is outlined in Algorithm 1. We explain each of

Algorithm 1: Progressively training DRL-based method.

Input: A non-parallel training example $X = (x_1, x_2, \cdots, x_n)$, a paraphrase generated by VAE $S = (s_1, s_2, \cdots, s_{m\prime})$, probability δ to pass VAE's output as input to decoder, probability ϵ to sample according to the policy, epoch number ω, pre-training status ρ, and the learning rate η.

1 Initialize $\mathcal{L}(\theta) \leftarrow 0$
2 **for** i=1,$\cdots$, m **do**
3 $vae\_in \leftarrow Uniform(0, 1)$
  if $vae\_in < \delta$ **then**
4 $\hat{y}_{i-1} \leftarrow s_{i-1}$
5 **if** $i \leq m - \omega$ OR $\rho = True$ **then**
6 $\hat{y}_i \leftarrow s_i$
7 **else**
8 $explore \leftarrow Uniform(0, 1)$
  if $explore < \epsilon$ **then**
9 $\hat{y}_i \leftarrow$ Sample $P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$
10 **else**
11 $\hat{y}_i \leftarrow$ Argmax $P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$
12 $\mathcal{L}(\theta) \leftarrow \mathcal{L}(\theta) + \log P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$
13 $\theta \leftarrow \theta + \eta \,.\, (\nabla\mathcal{L}(\theta) \,.\, r(X, \hat{Y}))$

the training phases below. Note that the pre-trained VAE and the DRL model share the same vocabulary.

Pre-train Phase: Pre-training is a critical step for DRL to work in practice. Since one of the main contributions of this work is to make DRL work in purely unsupervised fashion for the task of paraphrase generation, the pre-training step also has to be unsupervised. We use VAE [5], which is trained in an unsupervised way, and serves as a decent baseline in unsupervised paraphrase generation tasks [30]. The pre-trained VAE (section 2.2) guides as an expert in the pre-train phase to provide a warm-start. Line 6 in Algorithm 1 refers to the pre-train phase. At time-step i, the algorithm picks VAE's sample s_i as the action $\hat{y}_i$. The loss $\mathcal{L}(\theta)$ is computed and accumulated (see line 12 in Algorithm 1). Once, the action sequence is complete (i.e., $(\hat{y}_1, \hat{y}_2, \cdots, \hat{y}_m)$), the reward r is calculated and parameters θ are updated (line 13). This step is a requisite for the DRL model to work in practice for unsupervised paraphrasing.

Transition Phase: Once the model is able to generate sensible sentences, the next critical step is to progressively allow the agent (i.e., encoder-decoder model) to take actions according to its policy. Line 5 in Algorithm 1 refers to whether to take action according to the policy P_{DRL} or to utilize VAE's output S. First $max(0, m - \omega)$ tokens are picked from VAE, and the rest are sampled according to the policy $P_{DRL}(\hat{y}_i|h'_i, \hat{y}_{i-1})$ at time-step i, where m is the length of the generation (i.e., action sequence) and ω is the epoch number. This way, the model picks all tokens from VAE in epoch 0, and in epoch 1, the model is allowed to pick only the last token according to its policy, and so on. Similarly, by epoch m, the model learns to pick all the tokens according to its policy and none from the VAE. The intuition behind this gradual token-by-token transition is that mistakes at earlier tokens (i.e., words at the beginning of the sentence) can be catastrophic, and picking the last few tokens is relatively easy. Moreover, allowing the model to pick according to its policy as soon as possible is also needed, hence we employ gradual transitioning.

Since we allow the DRL model to pick according to its policy at an early stage in the transition phase, the model is expected to make mistakes. However, letting these errors compound over the next predictions may result in never being able to generate sufficiently good samples that can get high rewards. Lines 3-4 in Algorithm 1 attempt to overcome this issue by passing the VAE's previous token S_{i-1} to the decoder as input at time-step i with probability $\delta = sigmoid(m - i - \omega/l)) \in [0, 1]$, where m is the length of the output sentence, ω is the epoch number, and l is the slow-down factor to decay the probability δ as ω grows. It is similar to the above gradual transitioning, but l times slower and probabilistic. The intuition behind the slow progressive transition is that if the DRL model samples wrong token, passing the VAE's output to upcoming time-step's decoder would eliminate the accumulation of error in the beginning of the transition phase.

DRL Phase: The DRL phase is the classic reinforcement learning, where the agent takes action $\hat{Y}$ according to its policy P_{DRL}, gets reward r, and optimizes its policy to maximize its expected reward. Greedy decoding impedes the exploration of the space, whereas continuous exploring is also not a desirable behaviour. To keep a balance between exploration (i.e., sample) and exploitation (i.e., argmax), we use a probabilistic decaying mechanism for exploration with probability $\epsilon = \kappa^{\omega}$, where $\kappa \in [0, 1]$ is the constant to control the decay rate of the probability ϵ as ω grows. Lines 7-11 in Algorithm 1 refer to this phase. Pre-trained VAE is used as a baseline model in this phase.

Table 1: Statistics about paraphrase datasets

Dataset	Train	Valid	Test	Vocabulary
Quora	117K	3K	20K	8K
WikiAnswers	500K	6K	20K	8K
MSCOCO	110K	10K	40K	10k
Twitter	10K	2K	2K	8K

5 EXPERIMENTAL SETUP

In this section, we describe the datasets, competing approaches, evaluation metrics, and the implementation details of PUP.

5.1 Dataset

We use Quora [1], WikiAnswers [14], MSCOCO [25], and Twitter [22] datasets to evaluate the quality of the paraphrase generated by PUP and other competing approaches. Table 1 presents key statistics about the datasets. It is important to mention that although these datasets have parallel sentences, we don't use them for training nor for validation. We only use parallel sentences to compute the evaluation results on the respective testing sets.

Quora is a popular dataset for duplicate question detection annotated by humans which has been used for evaluating paraphrase quality as well, since a pair of duplicate questions can also be considered paraphrases of each other. We follow the training, validation,

Table 2: Performance of the unsupervised and domain-adapted methods on Quora and WikiAnswers datasets.

	Method	Quora				WikiAnswers			
		i-BLEU	BLEU	Rouge1	Rouge2	i-BLEU	BLEU	Rouge1	Rouge2
Supervised + domain adapted	Pointer-generator	5.04	6.96	41.89	12.77	21.87	27.94	53.99	20.85
	Transformer+Copy	6.17	8.15	44.89	14.79	23.25	29.22	53.33	21.02
	Shallow fusion	6.04	7.95	44.87	14.79	22.57	29.76	53.54	20.68
	MTL	4.90	6.37	37.64	11.83	18.34	23.65	48.19	17.53
	MTL+Copy	7.22	9.83	47.08	19.03	21.87	30.78	54.10	21.08
	DNPG	10.39	16.98	56.01	28.61	**25.60**	35.12	56.17	23.65
Unsupervised	VAE	8.16	13.96	44.55	22.64	17.92	24.13	31.87	12.08
	CGMH	9.94	15.73	48.73	26.12	20.05	26.45	43.31	16.53
	UPSA	12.02	18.18	56.51	**30.69**	24.84	32.39	54.12	21.45
	PUP	**14.91**	**19.68**	**59.77**	30.47	25.20	**38.22**	**58.88**	**26.72**

and testing splits used by [26, 30] for a fair comparison.

WikiAnswers contains 2M duplicate question-paraphrase pairs. We use $500K$ non-parallel sentences for training, following previous works [24, 26].

MSCOCO is an image captioning dataset that has over 120K images, each captioned by 5 different human annotators. Since all the captions for an image can be thought of as paraphrases, it has also been utilized for the paraphrasing task. We follow the standard splitting [25] and evaluation protocols [26, 34] in our experiments.

Twitter dataset is also annotated by humans for duplicate detection. We use the standard train/test split [22], and further split the training set to create a validation set (i.e., 2K sentences).

5.2 Baselines

We consider the following unsupervised baselines and domain-adapted approaches for comparison.

UPSA is a simulated annealing based approach [26] that attempts to generate paraphrases using a stochastic search algorithm and achieves state-of-art unsupervised paraphrasing results. We use its open source implementation to generate the paraphrases and compare against our approach.

CGMH is a Metropolis-Hastings based approach [30] that generates paraphrase by constraining the decoder at inference time. We use its open source implementation in our comparisons.

Domain-adapted models are trained in a supervised fashion on one dataset and adapted to another dataset in an unsupervised fashion. For this, we use previously reported results in [24] for Quora and WikiAnswers datasets.

We do not compare with the rule-based approaches such as [3, 28] due to the lack of availability of the rules or any implementation.

5.3 Evaluation Metrics

We use well-accepted automatic quantitative evaluation metrics as well as qualitative human studies in order to compare the performance of our method against the competing approaches. For quantitative measures, we use BLEU [32] and ROUGE [18] metrics, which have been widely utilized in the previous work to measure the quality of the paraphrases. Additionally, we use i-BLUE [41] by following the metrics in the most recent work [24, 26]. The metric i-BLUE [41] aims to measure the diversity of expression in the generated paraphrases by penalizing copying words from input sentences.

5.4 Implementation Details

The VAE contains two layers with 300-dimensional LSTM units. Our DRL-based model also has two-layers and uses 300-dimensional word embeddings (not pre-trained) and 300-dimensional hidden units. LSTM is utilized as a recurrent unit, and dropout of 0.5 is used. All the sentences are lower cased, and the maximum sentence length is 15 (i.e., we truncate longer sentences to maintain consistency with previous work). The vocabulary size for each dataset is listed in Table 1, and infrequent tokens are replaced with $< unk >$ token. We use Adam optimizer with learning rates of 0.15, 10^{-3}, and 10^{-4} in the pre-train, transition, and DRL phases, respectively. The mini-batch size is 32 and gradient clipping of a maximum gradient norm of 2 is used in all the phases. The validation is done after every epoch and the model with the best rewards is saved automatically. Whether to sample or use argmax, $\kappa = 0.9995$ is used. To compute the probability δ, which determines whether to pass VAE's output to the decoder, l is set to 8 during training. At inference time, we utilize beam search [47] with a beam size of $b = 8$ to sample paraphrases for the given input sentences. For the reward function, $\alpha = 0.4$, $\beta = 0.3$, $\gamma = 0.3$, $\tau_{min} = 0.3$, $\tau_{max} = 0.98$, and $\lambda_{min} = 0.3$ are used. All the hypterparameters are picked based on the validation split of the Quora dataset, and then consistently used for all the other datasets.

6 RESULTS

6.1 Automatic Metrics

Table 2 presents the performance of unsupervised and domain-adapted methods on the Quora and WikiAnswers datasets; the best method among all is shown in bold and the best among unsupervised methods is underlined for each metric. Unsupervised methods are trained with non-parallel corpora, and domain-adapted techniques are trained on Quora dataset in a supervised fashion and then domain adapted for WikiAnswers dataset in an unsupervised fashion (and vice versa). Our proposed method, PUP, outperforms all

Table 3: Performance of Unsupervised approaches on MSCOCO and Twitter dataset.

	MSCOCO				Twitter			
Method	i-BLEU	BLEU	Rouge1	Rouge2	i-BLEU	BLEU	Rouge1	Rouge2
VAE	7.48	11.09	31.78	8.66	2.92	3.46	15.13	3.4
CGMH	7.84	11.45	32.19	8.67	4.18	5.32	19.96	5.44
UPSA	9.26	14.16	37.18	11.21	4.93	6.87	28.34	8.53
PUP	**10.72**	**15.81**	**37.38**	**13.87**	**6.62**	**13.03**	**39.12**	**12.91**

Table 4: Subjective human studies on paraphrase generations by unsupervised methods on Quora dataset.

Method	Diversity	Fluency	Similarity
CGMH	3.14 ± 0.053	4.1 ± 0.042	2.97 ± 0.055
UPSA	2.96 ± 0.052	4.35 ± 0.033	3.89 ± 0.045
PUP	**3.27** ± 0.048	**4.42** ± 0.027	**4.09** ± 0.035

Table 5: Performance of the unsupervised methods for the components of the reward function on Quora dataset.

Method	Diversity	Fluency	Similarity	Reward
VAE	0.31	0.72	0.47	0.497
CGMH	0.29	0.73	0.49	0.502
UPSA	0.25	0.72	0.68	0.563
PUP	**0.53**	**0.95**	**0.81**	**0.768**

the unsupervised approaches on all metrics for Quora and WikiAnswers datasets (except Rouge2 for Quora dataset where performance is very competitive with UPSA). Similarly, PUP also outperforms domain-adapted methods for automatic metrics on Quora and WikiAnswers (except i-BLEU for WikiAnswers dataset where the performance is competitive). Although domain-adapted approaches have the advantage of supervised training on one dataset, this advantage does not transfer effectively to the other dataset despite the similarities between the datasets – i.e., Quora and WikiAnswers are both questions datasets. This also highlights that unsupervised approaches are worth exploring for the paraphrasing task as they can be applied to a variety of unlabeled domains or datasets in a flexible way without a need for adaptation. Moreover, the results for VAE (which we use to pre-train our DRL model) are presented in Table 2 and Table 3 to highlight the performance gain of PUP on each metric.

Table 3 presents the results of all unsupervised approaches on MSCOCO and Twitter datasets, where the best model is shown in bold for each metric. Our proposed method, PUP, is a clear winner on all the metrics among all the unsupervised approaches, which demonstrates the stellar performance of our method as well as the quality of our DRL reward function. The lower performance of unsupervised methods on Twitter dataset can be ascribed to the noisy tweets data, however, PUP has significantly better performance (i.e., 90% performance gain on BLEU, and 34% on i-BLEU scores with respect to UPSA) compared to other methods on all of the metrics, which signifies the robustness of the PUP.

6.2 Subjective Human Evaluations

To further illustrate the superior quality of the paraphrases generated by PUP, we conduct subjective human evaluations on Quora dataset. Table 4 presents the average scores along with the confidence intervals of human evaluators for diversity in expression, language fluency, and semantic similarity on randomly selected 300 paraphrases generated by all three unsupervised methods (CGMH, UPSA, and PUP). We used Amazon Mechanical Turk (a widely-used crowd sourcing platform) in our human studies. We selected Mechanical Turk *Masters* from the USA with a HIT approval rate of $\geq$ 90% to rate the paraphrases on a scale of $1-5$ (1 being the worst and 5 the best) for the three evaluation criteria diversity, fluency, and similarity. Each paraphrase is scored by three different evaluators. Our method PUP outperforms all the competing unsupervised approaches on all criteria. It should also be noted that CGMH is better on diversity of expression than UPSA, and the opposite results are observed for semantic similarity and fluency. In contrast, our reward function facilitates a good balance between the diversity in expression, semantic similarity, and fluency. A similar trend can also be observed in Table 5 and Table 6, which present automatically calculated reward and a few example paraphrases generated by all three unsupervised approaches, respectively.

6.3 Evaluation on Reward Function

Table 5 presents the average scores of all the components of our proposed reward function on Quora dataset for all the unsupervised approaches. Perhaps not surprisingly, our method outperforms other methods on each individual component of the reward by large margin. Intuitively, this arises from the fact that our DRL-based model is explicitly trained to optimize these reward components. Remarkably, DRL process improves the reward by more than 50% compared to the pre-training phase, i.e., the reward of VAE. This is also visible in Figure 3 where PUP starts with a reward value of around 0.5 and is able to achieve up to 0.77 towards the end of the last phase of training.

6.4 Ablation Study

Figure 3 presents the rewards achieved over the course of different epochs by three models: (*i*) PUP, pre-trained and uses the transition phase; (*ii*) `No Transition`, pre-trained but does not use the transition phase; and (*iii*) `No Pre-train`, not pre-trained at all. It highlights the need for the distinct phases in our training procedure. It can be observed that without the pre-training phase, `No Pre-train` model is unable to maximize the expected reward. The

Table 6: Example paraphrase generations by PUP and other unsupervised competing methods on Quora dataset.

Sr. #	Input Sentence	CGMH Generation	UPSA Generation	PUP Generation
1.	how can i work in microsoft	how can i prepare for cpt	how can i get to work at microsoft	how do i get a job at microsoft
2.	which is the best shampoo for dandruff	what is the best shampoo for sciatica	which is the best shampoo for oily skin	what are the proper shampoos for dandruff
3.	which book is the best to learn algo	which programming language is the best to learn algo	which book is best to learn algo	what is a best book for learning algos
4.	what is the best mac game	what is the best video game	what is the best mac app for android games	what are some good mac games
5.	what are the reasons of war	what are the positive aspects of nuclear war	what are the main reasons for a civil war	what is the main reason for war

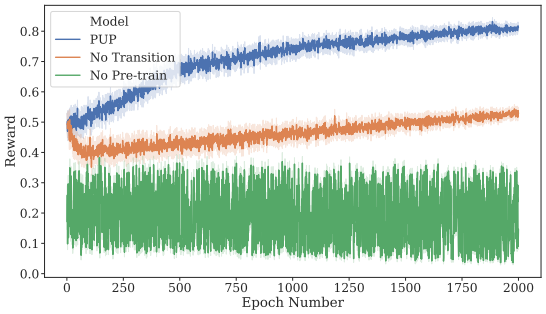

Figure 3: Evolution of the reward value for PUP variants over the course of the training.

reward remains small and fluctuates randomly. Similarly, transition phase is also required, as abrupt shift from VAE to DRL derails the training for `No Transition` model, whereas PUP is able to rapidly and consistently improve the reward as the number of epochs grow.

7 RELATED WORK

The automatic paraphrasing task is one of the common NLP tasks, which has widespread applications. A wide range of approaches were developed to solve this problem. Rule-based [3, 13, 28, 33] and data-driven approaches [27, 48] are some of the earliest techniques. Automatically constructed paraphrase detection datasets using SVM-based classifiers and other unsupervised approaches are introduced in [11, 12].

Recently, supervised deep learning approaches have also been used for paraphrase generation. Stacked residual LSTM networks [34] is one of the earliest efforts in the paraphrase generation utilizing deep networks. [23] makes use of deep reinforcement learning for paraphrase generation in a supervised fashion. Supervised paraphrase generation using LSTM-based VAE [15], transformer model [44], pointer-generator networks [37] have also shown promising results. Supervised paraphrase generation at different granularity levels (i.e., lexical, phrasal and sentential levels) [24] is achieved with template learning. Additionally these models can also be adapted to new domains in an unsupervised fashion, utilizing the learned templates with the assumption that both domains share similar templates.

Unsupervised paraphrasing is a challenging and emerging NLP task, and the literature is relatively limited. The VAE [5] is trained in an unsupervised fashion (i.e., no parallel corpus is required), by maximizing the lower bounds for the log-likelihood. The VAE's decoder can sample sentences (i.e., paraphrases), which are less controllable [30], but serve as a good baseline for the unsupervised paraphrasing task. CGMH [30] proposes a constrained sentence generation using Metropolis-Hastings Sampling by adding constraints on the decoder at inference time, and hence does not require parallel corpora. UPSA [26] generates paraphrases by simulated annealing, and achieves state-of-art results on the task. It proposes a search objective function, which involves semantic similarity and fluency for performing diverse word replacement, insertion or deletion operations, thus generating paraphrases in an unsupervised fashion. In contrast, we formulate the task as a deep reinforcement learning problem and progressively train the policy to maximize the expected reward, which includes semantic adequacy, language fluency, and diversity in expression.

8 CONCLUSION AND FUTURE WORK

We have presented a progressive approach to train a DRL-based unsupervised paraphrasing model. Our method provides a warm-start to the DRL-based model with a pre-trained VAE (i.e., trained on non-parallel corpus). Then, our model progressively transitions from VAE's output to acting according to its policy. We also propose a reward function which incorporates all the attributes of a good paraphrase and does not require parallel sentences. The paraphrases generated by our model outperform both state-of-the-art unsupervised paraphrasing and domain-adapted supervised models on automatic metrics. Specifically, our method achieves up to 90% and 34% performance gains for the BLEU and the i-BLEU metrics compared to state-of-the-art unsupervised methods, respectively. Moreover, the paraphrases generated by our method were rated the highest by human evaluators for all considered criteria: diversity of expression, fluency, and semantic similarity to input sentences. Since our technique is the first to successfully warm-start DRL with an unsupervised model, we plan on investigating the broader implications of our technique on other NLP problems with scarce labeled training data.

ACKNOWLEDGMENTS

This work is supported in part by the National Science Foundation (NSF) under grants IIS-1838222, IIS-1901379 and CNS-1932254.

REFERENCES

[1] [n.d.]. Quora Question Pairs | Kaggle. https://www.kaggle.com/c/quora-question-pairs. (Accessed on 02/14/2020).

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

[3] Regina Barzilay and Lillian Lee. 2003. Learning to paraphrase: an unsupervised approach using multiple-sequence alignment. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*. Association for Computational Linguistics, 16–23.

[4] Yoshua Bengio. 2008. Neural net language models. *Scholarpedia* 3, 1 (2008), 3881.

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

[6] Daniel Cer, Mona Diab, Eneko Agirre, Inigo Lopez-Gazpio, and Lucia Specia. 2017. Semeval-2017 task 1: Semantic textual similarity-multilingual and cross-lingual focused evaluation. *arXiv preprint arXiv:1708.00055* (2017).

[7] Daniel Cer, Yinfei Yang, Sheng-yi Kong, Nan Hua, Nicole Limtiaco, Rhomni St John, Noah Constant, Mario Guajardo-Cespedes, Steve Yuan, Chris Tar, et al. 2018. Universal sentence encoder. *arXiv preprint arXiv:1803.11175* (2018).

[8] David L Chen and William B Dolan. 2011. Collecting highly parallel data for paraphrase evaluation. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies-Volume 1*. Association for Computational Linguistics, 190–200.

[9] Hal Daumé, John Langford, and Daniel Marcu. 2009. Search-based structured prediction. *Machine learning* 75, 3 (2009), 297–325.

[10] Peter Dayan and Yael Niv. 2008. Reinforcement learning: the good, the bad and the ugly. *Current opinion in neurobiology* 18, 2 (2008), 185–196.

[11] Bill Dolan, Chris Quirk, and Chris Brockett. 2004. Unsupervised Construction of Large Paraphrase Corpora: Exploiting Massively Parallel News Sources. In *Proceedings of the 20th International Conference on Computational Linguistics* (Geneva, Switzerland) *(COLING '04)*. Association for Computational Linguistics, Stroudsburg, PA, USA, Article 350. https://doi.org/10.3115/1220355.1220406

[12] William B Dolan and Chris Brockett. 2005. Automatically constructing a corpus of sentential paraphrases. In *Proceedings of the Third International Workshop on Paraphrasing (IWP2005)*.

[13] Michael Ellsworth and Adam Janin. 2007. Mutaphrase: Paraphrasing with framenet. In *Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing*. Association for Computational Linguistics, 143–150.

[14] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. 2013. Paraphrase-driven learning for open question answering. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 1608–1618.

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

[16] Kenneth Heafield. 2011. KenLM: Faster and smaller language model queries. In *Proceedings of the sixth workshop on statistical machine translation*. Association for Computational Linguistics, 187–197.

[17] Sepp Hochreiter and Jürgen Schmidhuber. 1997. LSTM can solve hard long time lag problems. In *Advances in neural information processing systems*. 473–479.

[18] Eduard H Hovy, Chin-Yew Lin, Liang Zhou, and Junichi Fukumoto. 2006. Automated Summarization Evaluation with Basic Elements.. In *LREC*, Vol. 6. Citeseer, 899–902.

[19] Diederik P Kingma and Max Welling. 2013. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114* (2013).

[20] Emily Kissner. 2006. Summarizing, paraphrasing and retelling. *Portsmouth, NH: Heinernann* (2006).

[21] Kevin Knight and Daniel Marcu. 2000. Statistics-based summarization-step one: Sentence compression. *AAAI/IAAI* 2000 (2000), 703–710.

[22] Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A continuously growing dataset of sentential paraphrases. *arXiv preprint arXiv:1708.00391* (2017).

[23] Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2017. Paraphrase generation with deep reinforcement learning. *arXiv preprint arXiv:1711.00279* (2017).

[24] Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable neural paraphrase generation. *arXiv preprint arXiv:1906.09741* (2019).

[25] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[26] Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. 2019. Unsupervised Paraphrasing by Simulated Annealing. *arXiv preprint arXiv:1909.03588* (2019).

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

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

[29] Donald Metzler, Eduard Hovy, and Chunliang Zhang. 2011. An empirical evaluation of data-driven paraphrase generation techniques. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*. Association for Computational Linguistics, 546–551.

[30] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. 2019. Cgmh: Constrained sentence generation by metropolis-hastings sampling. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 6834–6842.

[31] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Alex Graves, Ioannis Antonoglou, Daan Wierstra, and Martin Riedmiller. 2013. Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602* (2013).

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

[33] Ellie Pavlick, Travis Wolfe, Pushpendre Rastogi, Chris Callison-Burch, Mark Dredze, and Benjamin Van Durme. 2015. Framenet+: Fast paraphrastic tripling of framenet. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*. 408–413.

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

[35] Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. 2014. Stochastic backpropagation and approximate inference in deep generative models. *arXiv preprint arXiv:1401.4082* (2014).

[36] Stéphane Ross, Geoffrey Gordon, and Drew Bagnell. 2011. A reduction of imitation learning and structured prediction to no-regret online learning. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*. 627–635.

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

[38] Pararth Shah, Dilek Hakkani-Tür, Gokhan Tür, Abhinav Rastogi, Ankur Bapna, Neha Nayak, and Larry Heck. 2018. Building a conversational agent overnight with dialogue self-play. *arXiv preprint arXiv:1801.04871* (2018).

[39] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *Nature* 550, 7676 (2017), 354.

[40] Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, Vol. 200.

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

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

[43] Richard S Sutton, Andrew G Barto, et al. 1998. *Introduction to reinforcement learning*. Vol. 135. MIT press Cambridge.

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

[45] Alex Warstadt, Amanpreet Singh, and Samuel R Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics* 7 (2019), 625–641.

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

[47] Sam Wiseman and Alexander M Rush. 2016. Sequence-to-sequence learning as beam-search optimization. *arXiv preprint arXiv:1606.02960* (2016).

[48] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. 2009. 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*. Association for Computational Linguistics, 834–842.

[49] Shiqi Zhao and Haifeng Wang. 2010. Paraphrases and applications. In *Coling 2010: Paraphrases and Applications–Tutorial notes*. 1–87.

[50] Shiqi Zhao, Haifeng Wang, Xiang Lan, and Ting Liu. 2010. Leveraging multiple MT engines for paraphrase generation. In *Proceedings of the 23rd International Conference on Computational Linguistics*. Association for Computational Linguistics, 1326–1334.

A SUPPLEMENTARY MATERIAL

A.1 Human Evaluations Details

A set of 300 randomly selected sentences from the test test of the Quora dataset were used for evaluation by crowd workers. The paraphrases generated by every model (i.e., CGMH, UPSA, PUP) were rated by three different crowd workers on the following criteria:

- **Semantic Similarity**: how close is the meaning of paraphrased sentence to the original sentence (i.e., 5 means same meaning, and 1 means completely different meaning).
- **Fluency**: whether the paraphrased sentence is grammatically acceptable (i.e., 5 means grammatically correct and 1 means that it makes no sense).
- **Diversity in expression**: whether different words are used in the paraphrased sentence with respect to the original sentence (i.e., 5 means at least half of the words are new, and 1 means that it makes no changes other than stop-words).

The raters were also provided with the positive (i.e., good example for each criteria) and negative (i.e., poor example for each criteria) examples for a sample sentence.

Test Sentence: To avoid carelessly filled responses, a test sentence (negative example) was placed with the three paraphrase generations (one from each model) for each input sentence, which was used to discard the rating provided by that particular worker for the paraphrases of that sentence. The workers were informed about the test sentence in the instructions. The responses of the workers who rated the sentence > 2 were discarded from the further analysis, which is reported in Section 6.2. However, workers were still paid.

There were a total of three test sentences; one of these was randomly placed in each set (three paraphrases by model, and one test sentence). The test sentence was easy to spot for: 1) totally different meaning than input (i.e., should get 1 on semantic similarity), 2) totally wrong for grammar correctness (i.e., should get 1 on fluency), and 3) same copy of the input (i.e., should get 1 on diversity in expression).

A.2 Datasets Preprocessing

We perform some of the standard pre-processing steps on all the datasets, which are briefly explained in the main paper as well. In this section, we explain the exact pre-processing steps. We use spaCyto tokenize the sentences. The maximum sentence length is set to 15, and longer sentences are trimmed (i.e., to remain consistent with previous works and easy comparison). We further pre-process and build vocabulary using torchtext by setting *init_token* (i.e., start of sentence) to <sos>, *eos_token* (i.e., end of sentence) to <eos>, and *lower* (i.e., lower case) to *True*. We also set *min_freq* (i.e., minimum frequency) to 4, *unk_init* (i.e.; infrequent/unknown token replacement) to <unk> for all the datasets, and we set *max_size* (i.e., vocabulary size) to 8K, 8K, 10K, and 8K for Quora, WikiAnswers, MSCOCO, and Twitter datasets respectively. No pre-trained word embeddings are utilized, instead embeddings are trained while models are being trained. Both the VAE, and DRL model share the same vocabulary.

A.3 Training Details

All the hyperparameters are described in Section 5.4. We follow the following steps to train the model for each dataset:

- The dataset is preprocessed as explained in Section A.2. All the datasets used for the experiments are publicly available.
- Variational Autoencoder (VAE) is trained for 15 epochs, which provides a warm-start to the our deep reinforcement learning-based model.
- The deep reinforcement learning-based unsupervised paraphrasing model is pre-trained for 15 epochs with the VAE.
- Then the model is trained in the transition, and DRL phases for 2000 epochs with the same parameters, as explained in Section 5.4.
- The weights for each component of the reward function, and the values for the thresholds are given in Section 5.4.
- The model that achieves best reward on the validation set is stored to generate paraphrases on the test-test for automatic evaluation metrics and human studies.