Abstract Rule Learning for Paraphrase Generation

Xianggen Liu , **Wenqiang Lei** , **Jiancheng Lv** and **Jizhe Zhou***
College of Computer Science, Sichuan University
{liuxianggen,lvjiancheng,jzzhou}@scu.edu.cn, wenqianglei@gmail.com

Abstract

In early years, paraphrase generation typically adopts rule-based methods, which are interpretable and able to make global transformations to the original sentence. But they struggle to produce fluent paraphrases. Recently, deep neural networks have shown impressive performances in generating paraphrases. However, the current neural models are black boxes and are prone to make local modifications to the inputs. In this work, we combine these two approaches into RULER, a novel approach that performs abstract *RUle LEaRning* for paraphrasing. The key idea is to explicitly learn generalizable rules that could enhance the paraphrase generation process of neural networks. In RULER, we first propose a rule generalizability metric to guide the model to generate rules underlying the paraphrasing. Then, we leverage neural networks to generate paraphrases by refining the sentences transformed by the learned rules. Extensive experimental results demonstrate the superiority of RULER over previous state-of-the-art methods in terms of paraphrase quality, generalization ability and interpretability.

1 Introduction

Paraphrase generation aims to restate the given sentence with different expressions while preserving the original semantics. It has been widely used in various downstream tasks such as information retrieval [Knight and Marcu, 2000], dialogue systems [Gao *et al.*, 2020], serving as a fundamental task in natural language processing (NLP).

In early years, paraphrasing was typically accomplished by knowledge-based or statistical machine translation (SMT) based approaches. Knowledge-based (also called rule-based) paraphrasing methods manually design explainable handcrafted rules [Barzilay and Lee, 2003] or automatically extract transforming patterns [Zhao *et al.*, 2009]. Though these paraphrasers provide robust out-of-domain quality, the generated sentences lack the fluency that readers usually expect. The SMT-based methods [Quirk *et al.*, 2004] formulate paraphrasing as a monolingual translation task based on sufficient parallel corpus. These SMT-based approaches are also easily explained but struggle to handle the sentences not seen before.

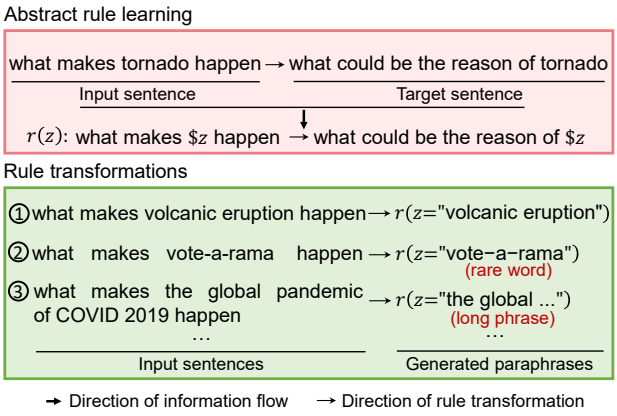

Figure 1: An example of the learning and transformation of an abstract rule r. $\$z$ is a symbol (placeholder) that can be assigned to various words during rule transformations.

Recently, neural networks have made impressive success in various NLP tasks, including paraphrase generation. Conventional neural-based approaches formulate paraphrasing as a supervised encoding-decoding problem [Wang *et al.*, 2019]. For instance, Prakash *et al.* [2016] use stacked residual LSTM networks to generate paraphrases. To improve the diversity of generated sentences, Gupta *et al.* [2018] introduce the variational auto-encoder (VAE) to perform paraphrase generation. But those generated paraphrases tend to only make trivial changes to original sentences, such as modifications from singular to plural. To be a good paraphrase, a sentence should share similar semantics but have noticeable syntactically or lexically differences from the original one.

To further push paraphrases lexically away from original sentences, SEPARATOR [Hosking and Lapata, 2021] first encodes the syntax and meaning separately and then decodes paraphrases by perturbing syntactic encodings. However, in such latent sampling procedures, the generated paraphrases are less "controllable" in the meaning preservation and lack interpretability of how the predictions are made.

In this work, we integrate the advantages of the rule-based and neural-based methods into RULER, an inter-

*Jizhe Zhou is the corresponding author

pretable framework that generates paraphrases by abstract *RUle LEaRning*. The abstract rule is the generalizable transformation from perceptual inputs to desired outputs. Our key idea is to learn abstract rules underlying paraphrasing that benefit the paraphrase generation process of neural networks. In particular, we first propose an evaluation metric to indicate the rule generalizability. Next, a rule learner learns to generate rules by maximizing the expected rewards (i.e., rule generalizability). Then, we select a suitable rule from the learned rules for the coming sentence and apply symbolic rule transformation. Finally, a neural network takes the transformed sentence as a reference to generate the paraphrase.

The advantage of performing paraphrasing by abstract rule learning is multi-fold: (1) Since rules are explicit knowledge (compared to neural networks), RULER presents good interpretability of why it produces such paraphrases. (2) RULER enjoys better generalizability to out-of-domain sentences due to symbolic rule transformations (Figure 1). (3) The rule transformation makes it possible to perform global modifications, which is a long-standing goal of paraphrase generation.

We evaluate the effectiveness of our method on two benchmark paraphrasing datasets, namely, Quora and Wikianswers. Experimental results show that RULER achieves a new state-of-the-art performance in terms of both automatic metrics and human evaluation. Moreover, we observe that RULER achieves better generalizability to the out-of-domain data.

2 Related Work

Traditional paraphrase generation methods mainly comprise rule-based methods [Mckeown, 1983; Barzilay and Lee, 2003; Zhao *et al.*, 2009; Lin and Pantel, 2001] and statistical machine translation (SMT) based methods [Quirk *et al.*, 2004]. In particular, Mckeown [1983] proposes to leverage rules extracted from the parsed grammatical tree to perform the transformations of given sentences. Lin and Pantel [2001] assume that dependency trees with similar arguments (leaves) are close in meaning. Based on the assumption, they extract paraphrasing rules to improve question answering. However, these rule-based systems can hardly ensure the fluency of the generated sentences. The SMT-based methods perform better in the quality of the generated paraphrases by calculating the translation probability from given inputs to the outputs [Dolan *et al.*, 2004].

With the development of deep learning, neural networks have become a prevailing approach to paraphrase generation, e.g., using VAE [Gupta *et al.*, 2018], residual LSTM [Prakash *et al.*, 2016] and the latent bag-of-words model [Fu *et al.*, 2019]. To improve the dissimilarity between the outputs and input sentences, Lin and Wan [2021] leverage multi-round paraphrase generation and back translation. Also, Liu *et al.* [2021] propose multi-round modification in hope of lexically different sequences (e.g., paraphrases). In addition, Li *et al.* [2019] generate paraphrases of a sentence at different levels of granularity in a disentangled way. However, the granularity of structures is hard to divide, which needs labor-intensive annotations to guide the learning.

Our work is also related to rule-enhanced neural networks. Hu *et al.* [2016] harness deep neural networks with a few manually-designed rules to perform text classification. But it could not be used in text generation and does not involve learning rules. To our knowledge, RULER is the first work of learning the transforming rules in paraphrase generation.

3 The Proposed Method

To learn to generate high-quality paraphrases, our method RULER consists of two processes, i.e., the abstract rule learning and the rule-enhanced generation, which will be elaborated in the rest of this section.

3.1 Abstract Rules in Paraphrase Generation

The abstract rule was first introduced to study the language acquisition of infants [Marcus *et al.*, 1999]. In this paper, we use an abstract rule to represent a type of paraphrase transformation. Let $\mathcal{W}$ be the word vocabulary and $\mathcal{Z}$ be the vocabulary of the symbols (placeholders) that can be assigned to various words. An abstract rule $r = \boldsymbol{w} \rightarrow \boldsymbol{v}$ is an implication corresponding to the conditional statement: "if $\boldsymbol{w}$ is matched, then produce $\boldsymbol{v}$", where $\boldsymbol{w} = [w_1, \ldots, w_{L_w}]$ and $\boldsymbol{v} = [v_1, \ldots, v_{L_v}]$ are the abstract patterns of the input sentence and the corresponding paraphrase, respectively. L_w and L_v are the lengths of tokens of $\boldsymbol{w}$ and $\boldsymbol{v}$. Note that $\boldsymbol{w}$ and $\boldsymbol{v}$ are not sentences because they commonly contain symbols.

Formally speaking, a rule r describes the mapping from the input sentence to its paraphrase, which is of the form

$$\begin{aligned} \boldsymbol{z} &= [z_1, \ldots, z_i, \ldots, z_{L_z}], z_i \in \mathcal{Z}, i \in 1, 2, \ldots, L_z \\ r(z) = \boldsymbol{w} &\rightarrow \boldsymbol{v} = [w_1, \ldots, w_{L_w}] \rightarrow [v_1, \ldots, v_{L_v}] \\ w_l, v_k &\in \mathcal{W} \cup \boldsymbol{z}; l \in 1, 2, \ldots, L_w; k \in 1, 2, \ldots, L_v, \end{aligned} \tag{1}$$

where L_z is the number of the symbols in rule r.

To apply rule r to transform a given sentence $\boldsymbol{X}$, the first step is to check whether the sentence $\boldsymbol{X}$ follows the pattern $\boldsymbol{w}$ and determine the words that the symbols $\boldsymbol{z}$ correspond to.

$$\hat{\boldsymbol{z}} = \text{Match}(r, \boldsymbol{X}), \tag{2}$$

where function "Match" performs token-wise alignment to identify the words in sentence $\boldsymbol{X}$ that match symbols in $\boldsymbol{z}$ (Figure 1). If sentence $\boldsymbol{X}$ is not matched to the rule, function "Match" returns a special token "`[None]`". Based on the assigned symbols $\hat{\boldsymbol{z}}$, the transformed sentence is

$$\boldsymbol{X}_r = r(\boldsymbol{z} = \hat{\boldsymbol{z}}), \tag{3}$$

where $r(\cdot)$ produces the sequence $\boldsymbol{v}$ but all symbols in $\boldsymbol{v}$ are assigned to the corresponding words. For notation simplicity, we use $\mathcal{T}(r, \boldsymbol{X})$ to represent the whole rule transformation.

$$\mathcal{T}(r, \boldsymbol{X}) = \boldsymbol{X}_r = r(\boldsymbol{z} = \text{Match}(r, \boldsymbol{X})). \tag{4}$$

3.2 Abstract Rule Learning

As shown in Figure 2(a), the rule learning process mainly involves a rule learner and a rule evaluation module. The rule learner takes a data point (including both the input sentence $\boldsymbol{X}$ and its paraphrase $\boldsymbol{Y}$) as input and generates a candidate rule underlying this data point. The rule evaluation module assesses the generalizability of the extracted rules, and thus guides the learning of the rule learner. The rest of this section will first formulate the rule generation subtask and then elaborate on these two components in detail.

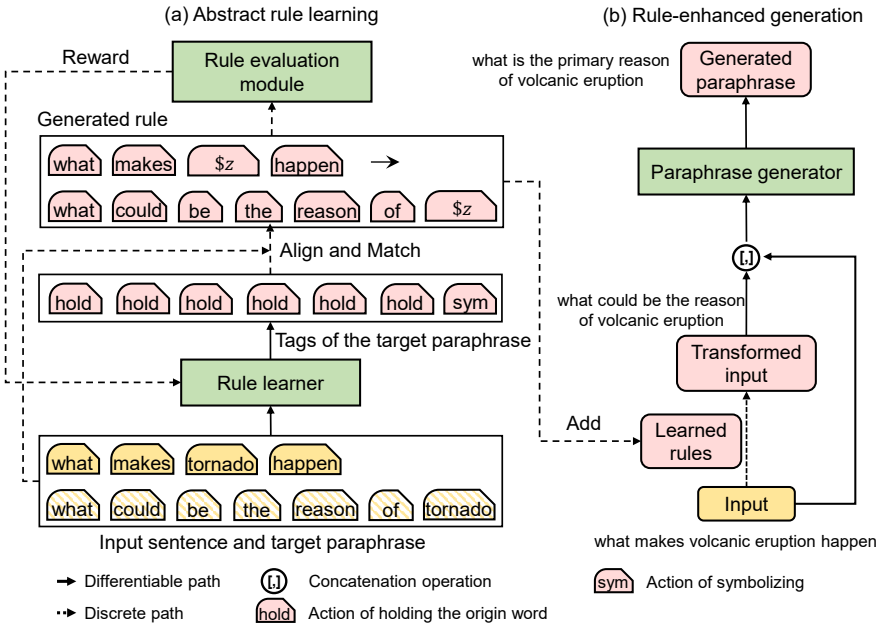

Figure 2: The overview of RULER framework.

Formulation of the Rule Generation. The rule generation subtask is to extract generalizable rules from a given data point. Let G be the rule learning agent. It functions as

$$r = G(\boldsymbol{X}, \boldsymbol{Y}). \tag{5}$$

However, if we generate the rule expression (i.e., Equation 1) token by token, the generative complexity is $\mathcal{O}(|\mathcal{W} \cup \mathcal{Z}|^{L+K})$, which will result in low search efficiency. Considering the evaluation of each rule is time-consuming (explained later), it will take a long time to figure out a suitable rule for a data point, making it intractable to train a rule learner. In this work, we try to simplify the rule generation subtask by restricting the output space of the rule learner.

Considering that the extracted rule r is expected to characterize the general mapping of the given data point, we assume that, the tokens of an abstract rule are either the words in the given data point or symbols. Specifically, for each word Y_l in the target paraphrase $\boldsymbol{Y}$, the rule learner predicts a tag O_l, deciding whether the word should be preserved (i.e., the action of holding) or symbolized (i.e., the action of symbolizing).

$$\boldsymbol{O} = [O_1, \ldots, O_{L_o}] = \text{RuleLearner}_{\boldsymbol{\theta}}(\boldsymbol{X}, \boldsymbol{Y}), \tag{6}$$

where $\boldsymbol{\theta}$ denotes all model parameters and $O_l \in \{\texttt{hold}, \texttt{symbolize}\}$. $\boldsymbol{O}$ is the output of the rule learner and L_o is the length of the output $\boldsymbol{O}$. Note that, to have a one-to-one mapping between the output and the target paraphrase, the length of the output is the same as the target paraphrase.

Based on the sequence of binary decisions of the rule learner, it is not difficult to align the symbols to the target paraphrase and the input sentence. Therefore, the abstract rule is obtained by

$$r = G(\boldsymbol{X}, \boldsymbol{Y}) = \text{AlignAndMatch}(\boldsymbol{O}, \boldsymbol{X}, \boldsymbol{Y}), \tag{7}$$

where "AlignAndMatch" stands for the computations from the output of the rule learner to the generated rule.

In this way, the generation complexity is reduced to $\mathcal{O}(2^{L_o})$, significantly lower than the original one.

Rule Learner. We use the standard Transformer architecture [Vaswani *et al.*, 2017] as the rule learner. Since both the input sentence and the target paraphrase are inputs of the rule learner, we insert a special separator token `[SEP]` between them and feed their concatenation into Transformer, given by

$$\text{RuleLearner}(\boldsymbol{X}, \boldsymbol{Y}) = \text{Transformer}([\boldsymbol{X}, [\texttt{SEP}], \boldsymbol{Y}]). \tag{8}$$

Rule Evaluation Module. Besides the huge search space, the lack of direct supervision is the second challenge of rule learning. In this work, we develop a rule evaluation module to automatically assess its quality. We consider the quality of a rule in two aspects, namely, the coverage of the rule and the usefulness of the rule for paraphrase generation.

First, the rule evaluation module tries to figure out how many data points that rule r covers. The wider coverage of a rule, the better generalization it possesses. Therefore, it calculates the coverage score by counting the samples from the training data that comply with the abstract rule r

$$\mathcal{D}_r = \{(\boldsymbol{X}, \boldsymbol{Y}) | \text{Match}(r, \boldsymbol{X}) \neq [\texttt{None}] \text{ and } (\boldsymbol{X}, \boldsymbol{Y}) \in \mathcal{D}\}, \tag{9}$$

where $\mathcal{D}$ is the training dataset and $\text{Match}(r, \boldsymbol{X}) \neq [\texttt{None}]$ denotes the event that sentence $\boldsymbol{X}$ follows rule r. Thus, $\mathcal{D}_r$ is the collection of matched samples of rule r.

In order to avoid enumerating all the training data, we stipulate a minimum number C_{min} of samples the rule should cover. If the matched samples are more than C_{min}, the coverage score is 1. Otherwise, it is the ratio of the number of matched samples to C_{min}, computed by

$$s_c(r) = \min\{\frac{|\mathcal{D}_r|}{C_{min}}, 1\}, \tag{10}$$

where $|\mathcal{D}_r|$ is the size of collection $\mathcal{D}_r$. Hence, for the evaluation of a rule, we only keep up to C_{min} samples in $\mathcal{D}_r$ to

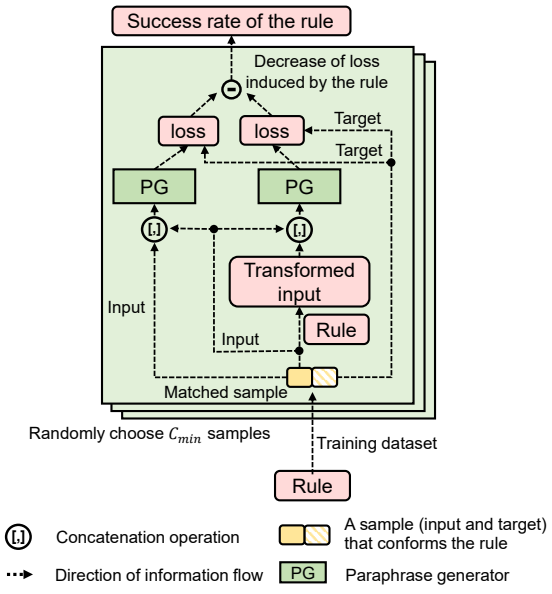

Figure 3: The architecture of the rule evaluation module.

accelerate the evaluation speed.

Second, the rule evaluation module tries to answer the question how many data points that rule r can help to perform paraphrasing. We suppose that, if the sentence transformed by rule r provides more information for paraphrase generation than the original sentence $\boldsymbol{X}$, rule r is useful to $\boldsymbol{X}$. As shown in Figure 3, the rule evaluation module leverages a trained paraphrase generator $\text{PG}_{\boldsymbol{\alpha}}$ to evaluate the difficulty of the paraphrasing based on the given inputs. For a rule r that is useful to sentence $\boldsymbol{X}$, we expect the output $\text{PG}_{\boldsymbol{\alpha}}(\boldsymbol{X}, \mathcal{T}(r, \boldsymbol{X}))$ is of better quality than the output $\text{PG}_{\boldsymbol{\alpha}}(\boldsymbol{X}, \boldsymbol{X})$. Considering one of the paraphrases of the input sentence $\boldsymbol{X}$ is available (i.e., $\boldsymbol{Y}$), the rule evaluation module adopts the reduction of the generation loss induced by the rule r to indicate the improvement of the generation:

$$\begin{aligned} s_{\text{redu}}(r, \boldsymbol{X}, \boldsymbol{Y}) =& \mathcal{L}(\text{PG}_{\boldsymbol{\alpha}}(\boldsymbol{X}, \boldsymbol{X}), \boldsymbol{Y}) \\ & - \mathcal{L}(\text{PG}_{\boldsymbol{\alpha}}(\boldsymbol{X}, \mathcal{T}(r, \boldsymbol{X})), \boldsymbol{Y}), \end{aligned} \tag{11}$$

where $\mathcal{L}$ is the cross-entropy loss of the model to generate target sentence $\boldsymbol{Y}$ from the inputs. $\boldsymbol{\alpha}$ denotes the parameters of $\text{PG}_{\boldsymbol{\alpha}}$. To eliminate the influence of noises, we impose a hyperparameter τ; If the reduction s_{redu} is greater than τ, we call rule r succeeds to help the paraphrase generation task.

The paraphrase generator is also implemented by a Transformer model, given by

$$\text{PG}_{\boldsymbol{\alpha}}(\boldsymbol{X}, \boldsymbol{X}_{aux}) = \text{Transformer}([\boldsymbol{X}, [\texttt{SEP}], \boldsymbol{X}_{aux}]), \tag{12}$$

where $\boldsymbol{X}_{aux}$ is the auxiliary input of $\text{PG}_{\boldsymbol{\alpha}}$, such as the sentences transformed by rules.

To reflect a statistical assessment of rule r in terms of usefulness, the rule evaluation module computes an average success rate of the r in helping the paraphrase generator over all the matched sentences

$$s_u(r) = \frac{\sum_{(\boldsymbol{X}, \boldsymbol{Y}) \sim \mathcal{D}_r} I_{\{s_{\text{redu}}(r, \boldsymbol{X}, \boldsymbol{Y}) > \tau\}}}{|\mathcal{D}_r|}, \tag{13}$$

which is used to indicate the usefulness of r. $I_{\{\}}$ is an indicator that yields 1 when its argument is true, and 0 otherwise.

Finally, the rule evaluation module combines the coverage score and the usefulness score into an overall score to represent the generalizability of the given rule, computed by

$$S(r) = s_c(r) \cdot s_u(r) \tag{14}$$

Optimization of Rule Learner. After determining the rule generation process and the evaluation of the rule, there is still a difficulty in optimizing the rule learner. That is, the labels contain only ultimate scores (i.e., $S(r)$), instead of full supervision to the each action. We therefore apply reinforcement learning to train our rule learner. Concretely, we define the generalizability score produced by the evaluation module as the reward of the rule learner. To maximize the expected reward, we compute the gradient of the policy, given by

$$\nabla_{\boldsymbol{\theta}} \mathcal{J} = \mathbb{E}_{\pi_{\boldsymbol{\theta}}} \nabla_{\boldsymbol{\theta}} S(r) \log \pi_{\boldsymbol{\theta}}(\boldsymbol{O} | \boldsymbol{X}, \boldsymbol{Y}), \tag{15}$$

where $\pi_\theta(\boldsymbol{O}|\boldsymbol{X}, \boldsymbol{Y})$ stands for the probability of generating the output $\boldsymbol{O}$ given data point $(\boldsymbol{X}, \boldsymbol{Y})$. We use REINFORCE algorithm [Williams, 1992] and follow the practical techniques in Liu *et al.* [2018] to train the rule learner.

When the abstract rule learning process is finished, we obtain a collection of the generated rules $\mathcal{R}$, given by

$$\mathcal{R} = \{r | S(r) > 0 \text{ and } r = G(\boldsymbol{X}, \boldsymbol{Y}) \text{ and } (\boldsymbol{X}, \boldsymbol{Y}) \in \mathcal{D}\}. \tag{16}$$

As the rules whose rewards are zeros have no contributions to paraphrase generation, they are not considered in $\mathcal{R}$.

3.3 Rule-enhanced Paraphrase Generation

For a sentence $\boldsymbol{X}$ with a suitable abstract rule r, we presume that it is easier to generate its paraphrase from the transformed sentence $\mathcal{T}(r, \boldsymbol{X})$ than the original sentence $\boldsymbol{X}$. Thus, we aim to train a powerful generator $\text{PG}_{\boldsymbol{\beta}}$ to perform paraphrasing based on the learned rules (Figure 2(b)), where $\boldsymbol{\beta}$ denotes the parameters of the paraphrase generator $\text{PG}_{\boldsymbol{\beta}}$.

We adopt same loss function for the paraphrase generator $\text{PG}_{\boldsymbol{\beta}}$ as used in the rule evaluator module, but with a different training set. In the rule-enhanced generation process, we add the transformed sentence as the auxiliary input into the training data of the paraphrase generator. Specifically, we select the matched rule with the highest reward to perform the transformation for each sample. Therefore, we have the rule-enhanced training dataset:

$$\begin{aligned} \mathcal{D}_g =& \{(\boldsymbol{X}, \mathcal{T}(r, \boldsymbol{X}), \boldsymbol{Y}) | (\boldsymbol{X}, \boldsymbol{Y}) \in \mathcal{D} \text{ and} \\ & r = \max_{r \in \mathcal{R}} (S(r) \cdot I_{\{\text{Match}(r, \boldsymbol{X}) \neq [\texttt{None}]\}})\}. \end{aligned} \tag{17}$$

When coming a new sentence $\hat{\boldsymbol{X}}$ for testing, we also perform the same procedures to obtain the auxiliary input (i.e., $\mathcal{T}(r, \hat{\boldsymbol{X}})$). Then we feed these two types of inputs into the trained $\text{PG}_{\boldsymbol{\beta}}$ to generate the final paraphrase $\hat{\boldsymbol{Y}}$, given by

$$\begin{aligned} \hat{r} &= \max_{r \in \mathcal{R}} (S(r) \cdot I_{\{\text{Match}(r, \hat{\boldsymbol{X}}) \neq [\texttt{None}]\}}) \\ \hat{\boldsymbol{Y}} &= \text{PG}_{\boldsymbol{\beta}}(\hat{\boldsymbol{X}}, \mathcal{T}(\hat{r}, \hat{\boldsymbol{X}})). \end{aligned} \tag{18}$$

4 Experiments

4.1 Datasets

We evaluate RULER on two widely used datasets, namely, the Quora question pairs and Wikianswers datasets. The

Model	Quora			Wikianswers		
	iBLEU	BLEU	METEOR	iBLEU	BLEU	METEOR
Copy	-7.17	**32.61**	60.06	-4.03	37.10	63.17
tf-idf [Hosking and Lapata, 2021]	-2.63	22.73	-	9.98	25.08	-
ParaNMT [Wieting and Gimpel, 2017]	0.04	24.24	-	2.32	20.42	-
SOW/REAP [Goyal and Durrett, 2020]	1.59	12.64	-	12.04	33.09	-
Autoencoder [Hosking and Lapata, 2021]	1.99	19.90	-	5.36	40.10	-
LBoW [Fu *et al.*, 2019]	2.62	16.17	47.28	13.71	34.96	47.36
VAE [Gupta *et al.*, 2018]	2.96	19.36	50.41	8.35	40.26	55.83
DiPS [Kumar *et al.*, 2019]	3.19	18.47	-	8.56	24.90	-
VQ-VAE [Van den Oord *et al.*, 2017]	3.43	16.19	-	8.47	40.26	-
SEPARATOR	5.48	14.35	45.14	14.84	36.36	49.81
HardMatch (degenerated RULER)	3.42	17.28	51.17	8.35	41.05	56.94
RULER	**5.86**	15.02	46.19	**15.07**	36.69	50.25

Table 1: Paraphrasing performance on the Quora and Wikianswers datasets. The best scores are highlighted in bold.

Model	Relevance		Dissimilarity	
	Mean Score	Agreement	Mean Score	Agreement
LBoW	3.28	0.39	3.19	0.50
VAE	3.31	0.42	2.56	0.55
SEPARATOR	2.97	0.48	**3.78**	0.54
RULER	**3.44**	0.45	3.50	0.49

Table 2: Human evaluation.

Quora dataset contains approximately 140K parallel sentences, which are sourced from question answering forum Quora. The Wikianswers dataset [Fader *et al.*, 2013] comprises 2.3M pairs of question paraphrases scraped from the Wikianswers website. On these two datasets, we adopt the same data splits with Hosking and Lapata [2021] for a fair comparison. Due to the complexity of rule learning, we limit our work to questions for the following two considerations: (1) The space of possible rules is smaller for questions than generic utterances, making the rule learning task more tractable; (2) It is more clearly defined for the concept of question paraphrases (i.e., corresponding to the same answer), thus leading to a better quality of the datasets.

4.2 Implementation Details

The paraphrase generators $\text{PG}_{\boldsymbol{\alpha}}$ and $\text{PG}_{\boldsymbol{\beta}}$ adopt the Transformer architecture and the previous best performing paraphraser model (i.e., SEPARATOR), respectively. To have a fair comparison with SEPARATOR, we use the same hyperparameters with it.

The other hyperparameters involved in our method include the minimum number C_{min} of the matched samples for a rule and the improvement threshold τ of the generator loss. As the training of the rule learner is time-consuming (approximately 8 days), these two hyperparameters were selected by manual tuning towards the paraphrasing performance on the validation set on Quora. The minimum number C_{min} of matched samples was set to 16. The improvement threshold τ of the generator loss is 0.2.

4.3 Competing Methods and Metrics

The competing methods used in our work include SEPARATOR [Hosking and Lapata, 2021], vector-quantized variational autoencoder (VQ-VAE) [Van den Oord *et al.*, 2017], diverse paraphraser using submodularity (DiPS) [Kumar *et al.*, 2019], and latent bag-of-words (LBoW) [Fu *et al.*, 2019], paraphraser by neural machine translation (ParaNMT) [Wieting and Gimpel, 2017], and neural syntactic preordering (denoted by SOW/REAP) [Goyal and Durrett, 2020].

Besides the neural-based approaches, we would compare RULER with the rule-based methods. However, the extracted rules of early work [Mckeown, 1983; Barzilay and Lee, 2003] were not available. We are unable to compare them in this paper. Therefore, we implement a rule-based control model named HardMatch. HardMatch shares the same framework with RULER but generates rules by the token-wise matching rather than the rule learner. The extracted rules of HardMatch are scored using iBLEU instead of $S(r)$. We also include a simple baseline named tf-idf. It retrieves the most similar sentences from the training set using tf-idf scores.

We adopt BLEU, iBLEU, and METEOR scores as automatic metrics to evaluate the generation performance. Among them, iBLEU is the BLEU score penalized by the similarity with the original sentence. It is more comprehensive for evaluation, and thus we take it as our major metric.

$$\text{iBLEU} = \lambda\text{BLEU}(\hat{\boldsymbol{Y}}, \hat{\mathcal{Y}}) - (1-\lambda)\text{BLEU}(\hat{\boldsymbol{Y}}, \hat{\boldsymbol{X}}), \qquad (19)$$

where $\hat{\mathcal{Y}}$ is the reference cluster of the input. We follow Hosking and Lapata [2021] to set λ to 0.7.

4.4 Results

Table 1 presents the performance of all competing methods on Quora and Wikianswers. We observe that, the Copy, tf-idf and ParaNMT methods show high BLEU scores but the lowest iBLEU scores, indicating that they are just parroting input sentences. The encoder-decoder frameworks, including Transformer, LBoW, VAE and SEPARATOR achieve higher iBLEU scores, which is mainly benefited from the supervised training data. However, such improvement comes with the reduction of the similarity with references. We further observe that RULER yields better results than the other paraphrasing systems, even the previous state-of-the-art model (i.e., SEPARATOR). In addition, the generation performance of HardMatch is also impressive, reflecting that the extracted rules by hard matching could also enhance the paraphrase generation process. But RULER can outperform it by a noticeable margin. This comparison shows that the rules underlying para-

		Wikianswers→Quora			Quora→Wikianswers		
	Model	iBLEU	BLEU	METEOR	iBLEU	BLEU	METEOR
Unsupervised	Copy	-7.17	**32.61**	**60.06**	-4.03	**37.10**	**63.17**
Domain adapted	CGMH [Miao *et al.*, 2019]	-1.08	15.79	44.35	-0.32	18.13	31.87
	UPSA [Liu *et al.*, 2020]	-0.33	17.85	47.82	0.28	20.83	33.25
No adaption	LBoW [Fu *et al.*, 2019]	1.24	7.05	27.16	3.71	16.87	29.03
	VAE [Gupta *et al.*, 2018]	1.29	7.28	27.23	0.13	28.41	49.83
	SEPARATOR [Hosking and Lapata, 2021]	1.78	6.28	27.03	6.09	17.32	29.76
	RULER	**2.57**	7.79	28.13	**6.15**	17.62	30.36

Table 3: Cross-domain performance from the Quora (or Wikianswers) dataset to the Wikianswers (or Quora) dataset.

Method	iBLEU	BLEU	METEOR
RULER	5.86	15.02	46.19
PG_β inference w/o rules transformation	5.69	14.87	45.79
PG_β training w/o rules	5.48	14.35	45.14

Table 4: Ablation study.

Input	How do i become an investment banker? What do they do
Matched rule	How do i become $\$z$? What do they do → What is the best way to be $\$z$
RULER	What is the best way to be an investment banker
SEPARATOR	How can i become an investment banker
Input	What is Taylor Swift 's nationality
Matched rule	What is $\$z$'s nationality → What nationality is $\$z$
RULER	What nationality is Taylor Swift
SEPARATOR	What is Taylor Swifts nationality and why
Input	Do wild animals suffer immensely when hunted by a wild predator, or are there mechanisms at work (such as adrenaline) that alleviate any suffering
Matched rule	`None`
RULER	Does wild animal suffer immensely when hunted by a wild predator, or are there any related mechanisms
SEPARATOR	Are wild animals more dangerous than wild animals

Table 5: Paraphrases generated by RULER and SEPARATOR.

phrasing can not be well extracted by simple matching.

It is curious to see how the paraphrase generators perform on a new domain where most sentences are not seen before. In this experiment, a supervised model is trained on one domain and directly tested on the other. As shown in Table 3, the generation performance of supervised methods decreases drastically on out-of-domain sentences. In addition, we also add two domain-adapted methods, including CGMH [Miao *et al.*, 2019] and UPSA [Liu *et al.*, 2020] into comparison. They are unsupervised paraphrasing methods but also perform adaption to the target domain. We find that our approach RULER still obtains better results than all the other methods. We conjecture this is because some of the extracted rules are shared across different domains, which makes RULER more generalizable to the out-of-domain data.

Human Evaluation. In addition to automatic evaluation, we ask three human annotators to evaluate the paraphrases generated by several competing generators (including SEPARATOR, VAE and LBoW). We use a total of 200 questions sampled equally from both Quora and Wikianswers. Each annotator is asked to evaluate generated paraphrases in terms of semantic relevance and expression dissimilarity, scoring from 1 to 5. Note that this experiment is conducted in a blind manner. The average human scores and Cohen's kappa scores are shown in Table 2. Most of the agreement scores are above 0.4, indicating moderate inter-annotator agreement. We observe RULER achieves the highest human satisfaction scores in terms of semantic relevance, and SEPARATOR performs better in expression dissimilarity. Compared with SEPARATOR, RULER presents a large advantage in semantic relevance and small drops in dissimilarity.

5 Model Analysis

Ablation Study. We then probe the influences of different design choices of the paraphrase generator to the performance of RULER. As shown in Table 4, if the generator PG_β uses the rules for training but does not use the rules to transform the input, the performance decreases by about 0.1 iBLEU. However, if we do not use any rules for training, the performance decreases heavily, suggesting the superiority of RULER mainly stems from the improvement of the quality of training data. We hope this observation provides a new perspective on how to use rules in neural networks.

Case Study and Error Analysis. We showcase several generated rules and corresponding paraphrases in Table 5. We see qualitatively that RULER produces more reasonable paraphrases than SEPARATOR in terms of both closeness in meaning and difference in expressions. In these examples, SEPARATOR just changes a few inconsequential words. By leveraging learned rules, RULER can make global modifications to the inputs. Meanwhile, RULER fails to provide suitable rules for some complicated sentences.

6 Conclusion and Future Work

Abstract rule learning is the ability to find generalizable transformations from the perceptual input to the desired output [Marcus *et al.*, 1999]. In this paper, we explore it in a specific problem, i.e., paraphrase generation. Our proposed method, RULER, first learns the abstract rules by reinforcement learning and then generates paraphrases by taking the rule-enhanced input as an auxiliary input. In the future, we plan to perform symbolic reasoning based on learned rules in hopes of generating more syntactically different sentences.

Acknowledgments

This work was supported by the State Key Program of the National Science Foundation of China under Grant 61836006.

References

[Barzilay and Lee, 2003] Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *ACL*, pages 16–23, 2003.

[Dolan *et al.*, 2004] William Dolan, Chris Quirk, Chris Brockett, and Bill Dolan. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *COLING*, pages 350–356, 2004.

[Fader *et al.*, 2013] Anthony Fader, Luke Zettlemoyer, and Oren Etzioni. Paraphrase-driven learning for open question answering. In *ACL*, pages 1608–1618, 2013.

[Fu *et al.*, 2019] Yao Fu, Yansong Feng, and John P Cunningham. Paraphrase generation with latent bag of words. In *NeurIPS*, pages 13645–13656, 2019.

[Gao *et al.*, 2020] Silin Gao, Yichi Zhang, Zhijian Ou, and Zhou Yu. Paraphrase augmented task-oriented dialog generation. In *ACL*, pages 639–649, 2020.

[Goyal and Durrett, 2020] Tanya Goyal and Greg Durrett. Neural syntactic preordering for controlled paraphrase generation. In *ACL*, pages 238–252, 2020.

[Gupta *et al.*, 2018] Ankush Gupta, Arvind Agarwal, Prawaan Singh, and Piyush Rai. A deep generative framework for paraphrase generation. In *AAAI*, pages 5149–5156, 2018.

[Hosking and Lapata, 2021] Tom Hosking and Mirella Lapata. Factorising meaning and form for intent-preserving paraphrasing. In *ACL-IJCNLP*, pages 1405–1418, 2021.

[Hu *et al.*, 2016] Zhiting Hu, Xuezhe Ma, Zhengzhong Liu, Eduard Hovy, and Eric Xing. Harnessing deep neural networks with logic rules. In *ACL*, pages 2410–2420, 2016.

[Knight and Marcu, 2000] Kevin Knight and Daniel Marcu. Statistics-based summarization step one: Sentence compression. In *AAAI*, pages 703–710, 2000.

[Kumar *et al.*, 2019] Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. Submodular optimization-based diverse paraphrasing and its effectiveness in data augmentation. In *NAACL*, pages 3609–3619, 2019.

[Li *et al.*, 2019] Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. Decomposable neural paraphrase generation. In *ACL*, pages 3403—3414, 2019.

[Lin and Pantel, 2001] Dekang Lin and Patrick Pantel. Discovery of inference rules for question-answering. *Natural Language Engineering*, 7(4):343–360, 2001.

[Lin and Wan, 2021] Zhe Lin and Xiaojun Wan. Pushing paraphrase away from original sentence: A multi-round paraphrase generation approach. In *ACL-IJCNLP*, pages 1548–1557, 2021.

[Liu *et al.*, 2018] Xianggen Liu, Lili Mou, Haotian Cui, Zhengdong Lu, and Sen Song. Jumper: Learning when to make classification decision in reading. In *IJCAI*, pages 4237–4243, 2018.

[Liu *et al.*, 2020] Xianggen Liu, Lili Mou, Fandong Meng, Hao Zhou, Jie Zhou, and Sen Song. Unsupervised paraphrasing by simulated annealing. In *ACL*, pages 302–312, 2020.

[Liu *et al.*, 2021] Xianggen Liu, Pengyong Li, Fandong Meng, Hao Zhou, Huasong Zhong, Jie Zhou, Lili Mou, and Sen Song. Simulated annealing for optimization of graphs and sequences. *Neurocomputing*, 465:310–324, 2021.

[Marcus *et al.*, 1999] Gary F Marcus, Sugumaran Vijayan, S Bandi Rao, and Peter M Vishton. Rule learning by seven-month-old infants. *Science*, 283(5398):77–80, 1999.

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

[Miao *et al.*, 2019] Ning Miao, Hao Zhou, Lili Mou, Rui Yan, and Lei Li. Constrained sentence generation by Metropolis–Hastings sampling. In *AAAI*, pages 6834–6842, 2019.

[Prakash *et al.*, 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*, pages 2923–2934, 2016.

[Quirk *et al.*, 2004] Chris Quirk, Chris Brockett, and William B Dolan. Monolingual machine translation for paraphrase generation. In *EMNLP*, pages 142–149, 2004.

[Van den Oord *et al.*, 2017] Aaron Van den Oord, Oriol Vinyals, et al. Neural discrete representation learning. In *NeurIPS*, pages 6306–6315, 2017.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, pages 5998–6008, 2017.

[Wang *et al.*, 2019] Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. A task in a suit and a tie: Paraphrase generation with semantic augmentation. In *AAAI*, pages 7176–7183, 2019.

[Wieting and Gimpel, 2017] John Wieting and Kevin Gimpel. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *ACL*, pages 451–462, 2017.

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

[Zhao *et al.*, 2009] Shiqi Zhao, Xiang Lan, Ting Liu, and Sheng Li. Application-driven statistical paraphrase generation. In *ACL*, pages 834–842, 2009.