Decomposable Neural Paraphrase Generation

Zichao Li, Xin Jiang, Lifeng Shang, Qun Liu
Huawei Noah's Ark Lab
{li.zichao, jiang.xin, shang.lifeng, qun.liu}@huawei.com

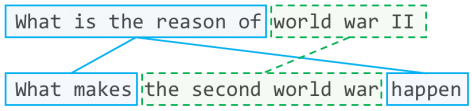
Abstract

Paraphrasing exists at different granularity levels, such as lexical level, phrasal level and sentential level. This paper presents Decomposable Neural Paraphrase Generator (DNPG), a Transformer-based model that can learn and generate paraphrases of a sentence at different levels of granularity in a disentangled way. Specifically, the model is composed of multiple encoders and decoders with different structures, each of which corresponds to a specific granularity. The empirical study shows that the decomposition mechanism of DNPG makes paraphrase generation more interpretable and controllable. Based on DNPG, we further develop an unsupervised domain adaptation method for paraphrase generation. Experimental results show that the proposed model achieves competitive in-domain performance compared to the state-of-the-art neural models, and significantly better performance when adapting to a new domain.

1 Introduction

Paraphrases are texts that convey the same meaning using different wording. Paraphrase generation is an important technique in natural language processing (NLP), which can be applied in various downstream tasks such as information retrieval, semantic parsing, and dialogue systems. Neural sequence-to-sequence (Seq2Seq) models have demonstrated the superior performances on generating paraphrases given a sentence (Prakash et al., 2016; Cao et al., 2017; Li et al., 2018; Ma et al., 2018). All of the existing works learn to paraphrase by mapping a sequence to another, with each word processed and generated in a uniform way.

This work is motivated by a commonly observed phenomenon that the paraphrase of a sentence is usually composed of multiple paraphrasing patterns at different levels of granularity, e.g., from the lexical to phrasal to sentential levels. For instance, the following pair of paraphrases contains both the phrase-level and the sentence-level patterns.



what is the reason of $x → *what makes $x happen*
world war II → the second world war

Specifically, the blue part is the sentence-level pattern, which can be expressed as a pair of sentence templates, where *$x* can be any fragment of text. The green part is the phrase-level pattern, which is a pair of phrases. Table 1 shows more examples of paraphrase pairs sampled from WikiAnswers corpus [1] and Quora question pairs [2]. We can see that the sentence-level paraphrases are more general and abstractive, while the word/phrase-level paraphrases are relatively diverse and domain-specific. Moreover, we notice that in many cases, paraphrasing can be decoupled, i.e., the word-level and phrase-level patterns are mostly independent of the sentence-level paraphrase patterns.

To address this phenomenon in paraphrase generation, we propose **D**ecomposable **N**eural **P**araphrase **G**enerator (DNPG). Specifically, the DNPG consists of a *separator*, multiple *encoders* and *decoders*, and an *aggregator*. The *separator* first partitions an input sentence into segments belonging to different granularities, which are then processed by multiple granularity-specific encoders and decoders in parallel. Finally the *aggregator* combines the outputs from all the decoders to produce a paraphrase of the input.

We explore three advantages of the DNPG:

[1] http://knowitall.cs.washington.edu/paralex/
[2] https://www.kaggle.com/c/quora-question-pairs

Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 3403–3414
Florence, Italy, July 28 - August 2, 2019.

Table 1: Examples of paraphrase pairs in WikiAnswers and Quora datasets. We manually labeled the sentences with the *blue italic words* being sentence-level and the green underlined words being phrase-level.

What is the population of New York*?* *How many people is there in* NYC*?*
Who wrote the Winnie the Pooh books*?* *Who is the author of* winnie the pooh*?*
What is the best phone *to buy below* 15k*?* *Which are* best mobile phones *to buy under* 15000*?*
How can I be a good geologist*?* *What should I do to be* a great geologist*?*
How do I reword a sentence *to* avoid plagiarism*?* *How can I* paraphrase my essay *and* avoid plagiarism*?*

Interpretable In contrast to the existing Seq2Seq models, we show that DNPG can automatically learn the paraphrasing transformation separately at lexical/phrasal and sentential levels. Besides generating a paraphrase given a sentence, it can meanwhile interpret its prediction by extracting the associated paraphrase patterns at different levels, similar to the examples shown above.

Controllable The model allows the user to control the generation process precisely. By employing DNPG, the user can specify the part of the sentence being fixed while the rest being rephrased at a particular level.

Domain-adaptable In this work, we assume that high-level paraphrase patterns are more likely to be shared across different domains. With all the levels coupled together, it is difficult for conventional Seq2Seq models to well adapt to a new domain. The DNPG model, however, can conduct paraphrase at abstractive (sentential) level individually, and thus be more capable of performing well in domain adaptation. Concretely, we develop a method for the DNPG to adapt to a new domain with only non-parallel data.

We verify the DNPG model on two large-scale paraphrasing datasets and show that it can generate paraphrases in a more controllable and interpretable way while preserving the quality. Furthermore, experiments on domain adaptation show that DNPG performs significantly better than the state-of-the-art methods. The technical contribution of this work is of three-fold:

1. We propose a novel Seq2Seq model that decomposes the paraphrase generation into learning paraphrase patterns at different granularity levels separately.

2. We demonstrate that the model achieves more interpretable and controllable generation of paraphrases.

3. Based on the proposed model, we develop a simple yet effective method for unsupervised domain adaptation.

2 Decomposable Neural Paraphrase Generator

This section explains the framework of the proposed DNPG model. We first give an overview of the model design and then elaborate each component in detail.

2.1 Model Overview

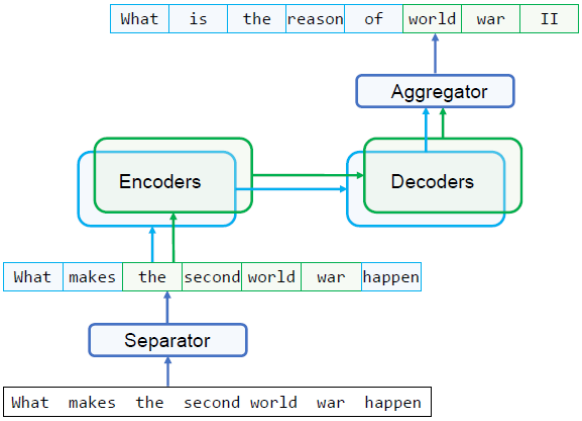


Figure 1: Model Architecture.

As illustrated in Figure 1, DNPG consists of four components: a separator, multi-granularity encoders and decoders (denoted as m-encoder and m-decoder respectively), and an aggregator. The m-encoder and m-decoder are composed of multiple independent encoders and decoders, with each corresponding to a specific level of granularity. Given an input sentence of words $X = [\mathrm{x}_1, \ldots, \mathrm{x}_L]$ with length L, the separator first determines the granularity label for each word, denoted as $Z = [z_1, \ldots, z_L]$. After that, the input sentence X together with its associated labels Z are fed into m-encoder in parallel and summarized as

$$U_z = m\text{-encoder}_z(X, Z), \quad (1)$$

where the subscript z denotes the granularity level. At the decoding stage, each decoder can individually predict the probability of generating the next

word y_t as

$$P_z(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X) = m\text{-decoder}_z(U_z, \mathrm{y}_{1:t-1}). \tag{2}$$

Finally, the aggregator combines the outputs of all the decoders and make the final prediction of the next word:

$$P(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X) = \sum_{z_t} P_{z_t}(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X)P(z_t|\mathrm{y}_{1:t-1}, X). \tag{3}$$

Here $P(z_t|\mathrm{y}_{1:t-1}, X)$ is computed as the probability of being at the granularity level z_t, and $P_{z_t}(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X)$ is given by the decoder $m\text{-decoder}_{z_t}$ at level z_t.

The choice of the encoder and decoder modules of DNPG can be quite flexible, for instance long-short term memory networks (LSTM) Hochreiter and Schmidhuber (1997) or convolutional neural network (CNN) (LeCun et al., 1998). In this work, the m-encoder and m-decoder are built based on the Transformer model (Vaswani et al., 2017). Besides, we employ LSTM networks to build the separator and aggregator modules. Without loss of generality, we consider two levels of granularity in our experiments, that is, $z = 0$ for the lexical/phrasal level and $z = 1$ for the sentential level.

2.2 Separator

For each word x_l in the sentence, we assign a latent variable z_l indicating its potential granularity level for paraphrasing. This can be simply formulated as a sequence labeling process. In this work we employ the stacked LSTMs to compute the distribution of the latent variables recursively:

$$\begin{aligned} h_l &= \text{BiLSTM}([\mathrm{x}_l; h_{l-1}, h_{l+1}]) \\ g_l &= \text{LSTM}([h_l, z_{l-1}; g_{l-1}]) \\ P(z_l|X) &= \text{GS}(W_g g_l, \tau) \end{aligned} \tag{4}$$

where h_l and g_l represent the hidden states in the LSTMs and $\text{GS}(\cdot, \tau)$ denotes the Gumbel-Softmax function (Jang et al., 2016). The reason of using Gumbel-Softmax is to make the model differentiable, and meanwhile produce the approximately discrete level for each token. τ is the temperature controlling the closeness of z towards 0 or 1.

2.3 Multi-granularity encoder and decoder

We employ the Transformer architecture for the encoders and decoders in DNPG. Specifically, the phrase-level Transformer is composed of m-encoder$_0$ and m-decoder$_0$, which is responsible for capturing the local paraphrasing patterns. The sentence-level Transformer is composed of m-encoder$_1$ and m-decoder$_1$, which aims to learn the high-level paraphrasing transformations. Based on the Transformer design in Vaswani et al. (2017), each encoder or decoder is composed of positional encoding, stacked multi-head attention, layer normalization, and feed-forward neural networks. The multi-head attention in the encoders contains self-attention while the one in the decoders contains both self-attention and context-attention. We refer readers to the original paper for details of each component. In order to better decouple paraphrases at different granularity levels, we introduce three inductive biases to the modules by varying the model capacity and configurations in the positional encoding and multi-head attention modules. We detail them hereafter.

Positional Encoding: We adopt the same variant of the positional encoding method in Vaswani et al. (2017), that is, the sinusoidal function:

$$\begin{aligned} \text{PE}(pos, 2d) &= \sin(p/10000^{2d/\mathbb{D}}) \\ \text{PE}(pos, 2d+1) &= \cos(p/10000^{2d/\mathbb{D}}) \end{aligned} \tag{5}$$

For phrase-level Transformer, we use the original position, i.e., $p := pos$. For the sentence-level Transformer, in order to make the positional encoding insensitive to the lengths of the phrase-level fragment, we set:

$$p = \sum_{i=1}^{pos} P(z_i = 1) \tag{6}$$

Multi-head Attention: We modify the self-attention mechanism in the encoders and decoders by setting a different receptive field for each granularity, as illustrated in Figure 2. Specifically, for the phrase-level model, we restrict each position in the encoder and decoder to attend only the adjacent n words ($n = 3$), so as to mainly capture the local composition. As for the sentence-level model, we allow the self-attention to cover the entire sentence, but only those words labeled as sentence-level (i.e., $z_l = 1$) are visible. In this manner, the model will focus on learning the sentence structures while ignoring the low-level details. To do so, we re-normalize the original atten-

tion weights $\alpha_{t,l}$ as

$$\alpha'_{t,l} = \frac{P(z_l = 1)\alpha_{t,l}}{\sum_{l=1}^{L} P(z_l = 1)\alpha_{t,l}}. \quad (7)$$

We also restrict the decoder at z level only access the position $l : z_l = z$ at encoder in the same way.

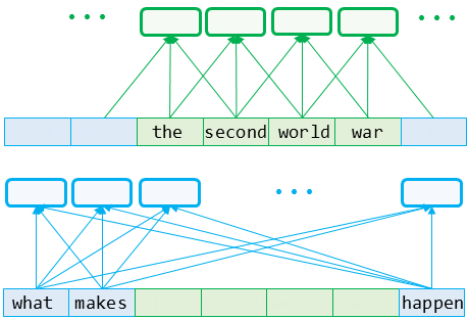


Figure 2: Attention: phrase-level self-attention (upper) and sentence-level self-attention (lower).

Model Capacity: We choose a larger capacity for the phrase-level Transformer over the sentence-level Transformer. The intuition behind is that lexical/phrasal paraphrases generally contain more long-tail expressions than the sentential ones. In addition, the phrase-level Transformer is equipped with the copying mechanism (Gu et al., 2016). Thus, the probability of generating the target word y_t by the m-decoder$_0$ is:

$$P_{z=0}(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X) = (1-\rho_t)P_{\text{gen}}(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X) + \rho_t P_{\text{copy}}(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X) \quad (8)$$

where ρ_t is the copying probability, which is jointly learned with the model. Table 2 summarizes the specifications of the Transformer models for each granularity.

Table 2: Model Specifications.

	Phrase-level model	Sentence-level model
Receptive field	Local	Global
Word Visibility	$\{x_l\}_{l=1}^{L}$	$\{\mathrm{x}_l\}_{l:z_l=1}$
#Dimension	300	150
#Heads	6	3
Copy mechanism	Yes	No

2.4 Aggregator

Each Transformer model works independently until generating the final paraphrases. The prediction of the token at t-th position is determined by the aggregator, which combines the outputs from the m-decoders. More precisely, the aggregator first decides the probability of the next word being at each granularity. The previous word y_{t-1} and the context vectors c_0 and c_1 given by m-decoder$_0$ and m-decoder$_1$, are fed into a LSTM to make the prediction:

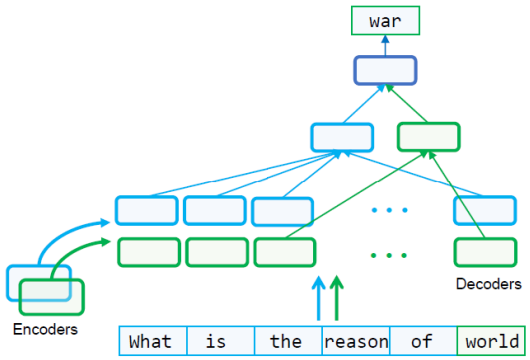


Figure 3: Aggregator.

$$v_t = \text{LSTM}([W_c[c_0; c_1; \mathrm{y}_{t-1}]; v_{t-1}])$$
$$P(z_t|\mathrm{y}_{1:t-1}, X) = \text{GS}(W_v v_t, \tau), \quad (9)$$

where v_t is the hidden state of the LSTM. Then, jointly with the probabilities computed by m-decoders, we can make the final prediction of the next word via Eq (3).

2.5 Learning of Separator and Aggregator

The proposed model can be trained end-to-end by maximizing the conditional probability (3). However, learning from scratch may not be informative for the separator and aggregator to disentangle the paraphrase patterns in an optimal way. Thus we induce weak supervision to guide the training of the model. We construct the supervision based on a heuristic that long-tail expressions contain more rare words. To this end, we first use the word alignment model (Och and Ney, 2003) to establish the links between the words in the sentence pairs from the paraphrase corpus. Then we assign the label $z^* = 0$ (phrase-level) to n (randomly sampled from $\{1, 2, 3\}$) pairs of aligned phrases that contain most rare words. The rest of the words are labeled as $z^* = 1$ (sentence-level).

We train the model with explicit supervision at

the beginning, with the following loss function:

$$\mathcal{L} = \sum_{t=1}^{T} \log P(\mathrm{y}_t|\mathrm{y}_{1:t-1}, X) + \lambda(\sum_{l=1}^{L} \log P(z_l^*|X) + \sum_{t=1}^{T} \log P(z_t^*|\mathrm{y}_{1:t-1}, X)) \quad (10)$$

where λ is the hyper-parameter controlling the weight of the explicit supervision. In experiments, we decrease λ gradually from 1 to nearly 0.

3 Applications and Experimental Results

We verify the proposed DNPG model for paraphrase generation in three aspects: interpretability, controllability and domain adaptability. We conduct experiments on WikiAnswers paraphrase corpus (**?**) and Quora duplicate question pairs, both of which are questions data. While the Quora dataset is labeled by human annotators, the WikiAnswers corpus is collected in a automatic way, and hence it is much noisier. There are more than 2 million pairs of sentences on WikiAnswers corpus. To make the application setting more similar to real-world applications, and more challenging for domain adaptation, we use a randomly sampled subset for training. The detailed statistics are shown in Table 3.

Table 3: Statistics of the paraphrase datasets.

	WikiAnswers	Quora
Training	500K	100K
Validation	6K	4K
Test	20K	20K

3.1 Implementation and Training Details

As the words in the WikiAnswers are all stemmed and lower case, we do the same pre-processing on Quora dataset. For both datasets, we truncate all the sentences longer than 20 words. For the models with copy mechanism, we maintain a vocabulary of size 8K. For the other baseline models besides vanilla Transformer, we include all the words in the training sets into vocabulary to ensure that the improvement of our models does not come from solving the out-of-vocabulary issue. For a fair comparison, we use the Transformer model with similar number of parameters with our model. Specifically, it is with 3 layers, model size of 450 dimensions, and attention with 9 heads. We use early stopping to prevent the problem of over-fitting. We train the DNPG with Adam optimizer (Kingma and Ba, 2014). We set the learning rate as $5e-4$, τ as 1 and λ as 1 at first, and then decrease them to $1e-4$, 0.9 and $1e-2$ after 3 epochs. We set the hyper-parameters of models and optimization in all other baseline models to remain the same in their original works. We implement our model with PyTorch (Paszke et al., 2017).

3.2 Interpretable Paraphrase Generation

First, we evaluate our model quantitatively in terms of automatic metrics such as BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), which have been widely used in previous works on paraphrase generation. In addition, we include iBLEU (Sun and Zhou, 2012), which penalizes repeating the source sentence in its paraphrase. We use the same hyper-parameter in their original work. We compare DNPG with four existing neural-based models: ResidualLSTM (Prakash et al., 2016), VAE-SVG-eq (Gupta et al., 2017), pointer-generator (See et al., 2017) and the Transformer (Vaswani et al., 2017), the latter two of which have been reported as the state-of-the-art models in Li et al. (2018) and Wang et al. (2018) respectively. For a fair comparison, we also include a Transformer model with copy mechanism. Table 4 shows the performances of the models, indicating that DNPG achieves competitive performance in terms of all the automatic metrics among all the models. In particular, the DNPG has similar performance with the vanilla Transformer model on Quora dataset, while significantly performs better on WikiAnswers. The reason maybe that the DNPG is more robust to the noise, since it can process the paraphrase in an abstractive way. It also validates our assumption that paraphrasing can be decomposed in terms of granularity. When the training data of high quality is available, the transformer-based models significantly outperforms the LSTM-based models.

Besides the quantitative performance, we demonstrate the interpretability of DNPG. Given an input sentence, the model can not only generate its paraphrase but also predict the granularity level of each word. By using the predicted granularity levels and the context attentions in the Transformer, we are able to extract the phrasal and sentential paraphrase patterns from the model. Specifically, we extract the sentential templates $\bar{X}$

Table 4: In-domain performance of paraphrase generation.

	Quora				WikiAnswers			
Models	BLEU	iBLEU	ROUGE-1	ROUGE-2	BLEU	iBLEU	ROUGE-1	ROUGE-2
ResidualLSTM	17.57	12.67	59.22	32.40	27.36	22.94	48.52	18.71
VAE-SVG-eq	20.04	15.17	59.98	33.30	32.98	26.35	50.93	19.11
Pointer-generator	22.65	16.79	61.96	36.07	39.36	31.98	57.19	25.38
Transformer	21.73	16.25	60.25	33.45	33.01	27.70	51.85	20.70
Transformer+Copy	24.77	17.98	63.34	37.31	37.88	31.43	55.88	23.37
DNPG (ours)	**25.03**	**18.01**	**63.73**	**37.75**	**41.64**	**34.15**	**57.32**	**25.88**

Table 5: Performance of paraphrase generation on domain adaptation (source → target).

	WikiAnswers→ Quora				Quora→WikiAnswers			
Models	BLEU	iBLEU	ROUGE-1	ROUGE-2	BLEU	iBLEU	ROUGE-1	ROUGE-2
Pointer-generator	6.96	5.04	41.89	12.77	27.94	21.87	53.99	20.85
Transformer+Copy	8.15	6.17	44.89	14.79	29.22	23.25	53.33	21.02
DNPG (ours)	10.00	7.38	47.53	18.89	31.84	24.22	54.87	22.27
Shallow fusion	7.95	6.04	44.87	14.79	29.76	22.57	53.54	20.68
MTL	6.37	4.90	37.64	11.83	23.65	18.34	48.19	17.53
MTL+Copy	9.83	7.22	47.08	19.03	30.78	21.87	54.10	21.08
Adapted DNPG (ours)	**16.98**	**10.39**	**56.01**	**28.61**	**35.12**	**25.60**	**56.17**	**23.65**

of X (or $\bar{Y}$ of Y) by substituting each fragment of words at the phrasal level by a unique placeholder such as *$x*. The extraction process is denoted as $\bar{X} = \mathcal{T}(X, Z) = [\bar{\mathrm{x}}_1, \ldots, \bar{\mathrm{x}}_{\bar{L}}]$, where the element $\bar{\mathrm{x}}_{\bar{l}}$ is either a placeholder or a word labeled as sentence-level. Through the attention weights, we ensure that the pair of aligned fragment share the same placeholder in $\{\bar{X}, \bar{Y}\}$. The whole generation and alignment process is detailed in Appendix A. Each pair of fragments sharing the same placeholder are extracted as the phrasal paraphrase patterns.

Table 6 gives examples of the generated paraphrases and the corresponding extracted templates. For instance, the model learns a sentential paraphrasing pattern: $\bar{X}$: *what is $x's $y* → $\bar{Y}$: *what is the $y of $x*, which is a common rewriting pattern applicable in general practice. The results clearly demonstrate the ability of DNPG to decompose the patterns at different levels, making its behaviors more interpretable.

3.3 Controllable Paraphrase Generation

The design of the DNPG model allows the user to control the generating process more precisely. Thanks to the decomposable mechanisms, it is flexible for the model to conduct either sentential paraphrasing or phrasal paraphrasing individually. Furthermore, instead of using the learned separator, the user can manually specify the granularity labels of the input sentence and then choose the following paraphrasing strategies.

Sentential paraphrasing is performed by restricting the phrase-level decoder (m-decoder$_0$) to copying from the input at the decoding stage, i.e., keeping the copy probability $\rho_t = 1$. To ensure that the phrasal parts are well preserved, we replace each phrase-level fragment by a unique placeholder and recover it after decoding.

Phrasal paraphrasing is performed with sentence template being fixed. For each phrase-level fragment, paraphrase is generated by m-decoder$_0$ only and the generation stopped at $t : z_t = 1$.

Once the beam search of size B finished, there are B paraphrase candidates $\hat{Y}_b$. We pick up the one with the best accuracy and readability. Specifically, we re-rank them by $P(\hat{Y}_b|X, Z)$ calculated by the full model of DNPG.

Given a sentence, we manually label different segments of words as phrase-level, and employ the model to conduct sentential and phrasal paraphrasing individually. With the manual labels, the model automatically selects different paraphrase patterns for generation. Table 7 shows examples of the generated results by different paraphrasing strategies. As demonstrated by the examples, DNPG is flexible enough to generate paraphrase given different sentence templates and phrases.

Controllable generation is useful in downstream applications, for instance, data augmentation in the task-oriented dialogue system. Suppose we have the user utterance *book a flight from* New York *to* London and want to produce more

Table 6: Examples of the generated paraphrases and extracted patterns at each granularity level by DNPG.

Input Sentence	Generate Paraphrase	Sentential Paraphrase Patterns	Phrasal Paraphrase Patterns
what is the technique for prevent suicide?	how can you prevent suicide?	*what is the technique for $x* → *how can you $x*	-
what is the second easiest island?	what is the 2nd easiest island?	-	second easiest island → 2nd easiest island
what is rihanna brother's name?	what is the name of rihanna's brother?	*what is $x's $y* → *what is the $y of $x*	rihanna brother → rihanna's brother
do anyone see the relation between greek god and hindu god?	what is the relationship between the greek god and hindu god?	*do anyone see the $x between $y* → *what is the $x between the $y*	relation → relationship

Table 7: Examples of controllable generation of paraphrase. The words with underline are labeled as phrase-level and the ones in *italic* form are at sentence-level. The strategy *All* is referred as the fully automatic generation.

Input sentence & labels	Strategy	Generated Paraphrase
what is the value of a 1961 us cent?	All	what is the 1961 nickel 's value?
what is the value *of* a 1961 us cent*?*	Phrase	*what is the* price *of* a 1961 nickel*?*
what is the value *of* a 1961 us cent*?*	Sentence	*what is the* 1961 us cent *'s* value*?*
what is the value of a 1961 us cent*?*	Phrase	*what is the value of* a 1961 nickel*?*
what is the value of a 1961 us cent*?*	Sentence	*how much is* a 1961 us cent *worth?*
what should i do to avoid sleep in class?	All	how do i prevent sleep in class?
what should i do to avoid sleep in class*?*	Phrase	*what should i do to* prevent sleep in class*?*
what should i do to avoid sleep in class*?*	Sentence	*how do i* avoid sleep in class*?*
what should i do to avoid sleep in class*?*	Phrase	*what should i do to avoid* fall sleep during class*?*
what should i do to avoid sleep in class*?*	Sentence	*what should i do if i don't want to* sleep in class*?*

utterances with the same intent. With the DNPG, we can conduct sentential paraphrasing and keep the slot values fixed, e.g. *buy an airline ticket to* London *from* New York.

3.4 Unsupervised Domain Adaptation

Existing studies on paraphrase generation mainly focus on the in-domain setting with a large-scale parallel corpus for training. In practice, there is always a need to apply the model in a new domain, where no parallel data is available. We formulate it as an unsupervised domain adaptation problem for paraphrase generation. Based on the observation that the sentence templates generated by DNPG tend to be more general and domain-insensitive, we consider directly performing the sentential paraphrase in the target domain as a solution. However, the language model of the source and target domains may differ, we therefore fine-tune the separator of DNPG so that it can identify the granularity of the sentence in the target domain more accurately. Specifically, to adapt the separator $P_{\text{sep}}(Z|X)$ to the target domain, we employ a reinforcement learning (RL) approach by maximizing the accumulative reward:

$$\mathcal{R}_{\text{separator}} = \mathbb{E}_{P_{\text{sep}}(Z|X)} \sum_{l=1}^{L} r_l(z_{1:l}, X). \quad (11)$$

We define the reward functions based on the principle that the source and target domain share the similar sentence templates. We first train a neural language model, specifically LSTM, with the sentence templates in the source domain, with the conditional probability denoted as $P_{\text{LM}}(\bar{\text{x}}_{\bar{l}}|\bar{\text{x}}_{1:\bar{l}-1})$. In the target domain, the template language model is employed as a reward function for separator. Formally we define the reward r_l at position l as:

$$r_l(z_{1:l}, X) = \alpha P_{\text{LM}}(\bar{\text{x}}_{\bar{l}}|\bar{\text{x}}_{1:\bar{l}-1}), \quad (12)$$

where the template $\bar{\text{x}}_{1:\bar{l}} = \mathcal{T}(X, z_{1:l})$ is extracted in the way as detailed in Section 3.2. And α is a scaling factor that penalizes the long fragment labeled as phrase-level, since more informative sentence templates are preferred. With the reward, the separator is further tuned with the policy gradient method (Williams, 1992; Sutton et al., 2000). To bridge the gap between training and testing of the Transformer models in different domain, we fine-tune the DNPG model on the sentential paraphrase patterns extracted in source domain. Since only the unlabeled data in the target domain is needed to fine-tune separator, the domain adaptation can be done incrementally. An overview of the complete training process is illustrated in Figure 4. We refer the model fine-tuned in this way as Adapted DNPG.

We evaluate the performance of the original DNPG and the Adapted DNPG in two settings of domain transfer: 1) Quora dataset as the source domain and WikiAnswers as the target domain, denoted as Quora→WikiAnswers, and 2) in reverse as WikiAnswers→Quora. For the baseline models, in addition to the pointer-generator network and the Transformer model with copy mechanism (denoted as Transformer+Copy), we use

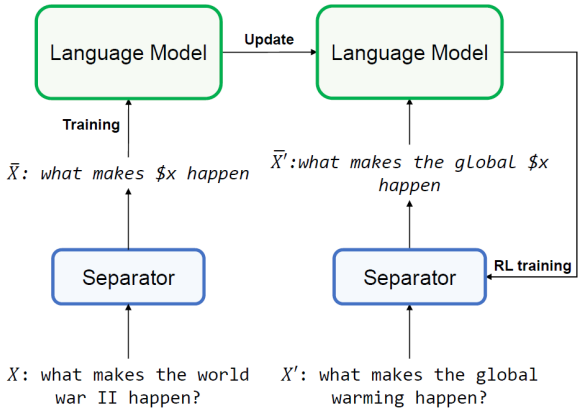


Figure 4: Left: Training of language model in the source domain; Right: RL training of separator in the target domain.

the shallow fusion (Gulcehre et al., 2015) and the multi-task learning (MTL) (Domhan and Hieber, 2017) that harness the non-parallel data in the target domain for adaptation. For fair comparisons, we use the Transformer+Copy as the base model for shallow fusion and implement a variant of MTL with copy mechanism (denoted as MTL+Copy). Table 5 shows performances of the models in two settings. DNPG achieves better performance over the pointer-generator and Transformer-based model, and has the competitive performance with MTL+Copy, which accesses target domain for training. With a fine-tuned separator, Adapted DNPG outperforms other models significantly on Quora→WikiAnswers. When it comes to WikiAnswers→Quora, where domain adaptation is more challenging since the source domain is noisy, the margin is much larger. The main reason is that the original meaning can be preserved well when the paraphrasing is conducted at the sentential level only. For an intuitive illustration, We show examples of the generated paraphrases from Adapted DNPG and MTL+Copy in Table 10 in Appendix. It is shown that the sentential paraphrasing is an efficient way to reuse the general paraphrase patterns and meanwhile avoid mistakes on rephrasing domain-specific phrases. However, it is at the expense of the diversity of the generated paraphrases. We leave this problem for future work.

To further verify the improvement of Adapted DNPG, we conduct the human evaluation on the WikiAnswers→Quora setting. We have six human assessors to evaluate 120 groups of paraphrase candidates given the input sentences. Each group consists of the output paraphrases from

Table 8: Human Evaluation in WikiAnswers→Quora

Models	Mean Rank	Agreement
MTL+Copy	3.22	0.446
Naive DNPG	3.13	0.323
Adapted DNPG	**1.79**	0.383
Reference	1.48	0.338

MTL+Copy, DNPG and Adapted DNPG as well as the reference. The evaluators are asked to rank the candidates from 1 (best) to 4 (worst) by their readability, accuracy and surface dissimilarity to the input sentence. The detailed evaluation guide can be found in Appendix B. Table 8 shows the mean rank and inter-annotator agreement (Cohen's kappa) of each model. Adapted DNPG again significantly outperforms MTL+Copy by a large margin (p-value < 0.01). The performance of the original DNPG and MTL+Copy has no significant difference (p-value $= 0.18$). All of the inter-annotator agreement is regarded as *fair* or above.

3.5 Ablation Studies and Discussion

We quantify the performance gain of each inductive bias we incorporated in the DNPG model. Specifically, we compare the DNPG with three variants: one with vanilla attention modules, one with vanilla positional encoding and the one uses vanilla softmax. We train them with the training set of WikiAnswers and test in the validation set of Quora. The results are shown in Table 9, which shows that each inductive bias has a positive contribution. It further proves that the decomposition mechanism allows the model to capture more abstractive and domain-invariant patterns. We also note that there is a large drop without the constraints on multi-head attention, which is a core part of the decomposition mechanism. We investigate the effect of the weak supervision for separator and aggregator by setting λ as 0. Though there is not a significant drop on quantitative performance, we observe that the model struggles to extract meaningful paraphrase patterns. It means that explicit supervision for separator and aggregator can make a difference and it does not need to be optimal. It opens a door to incorporate symbolic knowledge, such as regular expression of sentence templates, human written paraphrase patterns, and phrase dictionary, into the neural network. Through training in a parallel corpus, DNPG can generalize the symbolic rules.

Table 9: Ablation Study in WikiAnswers→Quora

Model Variants	BLEU	iBLEU
DNPG	9.84	7.40
w/ Vanilla Multi-Head Attention	7.65	5.86
w/ Vanilla Positional Encoding	9.46	7.08
w/ Vanilla Softmax	9.30	7.01

4 Related Work

4.1 Neural Paraphrase Generation

Most of the existing neural methods of paraphrase generation focus on improving the in-domain quality of generated paraphrases. Prakash et al. (2016) and Ma et al. (2018) adjust the network architecture for larger capacity. Cao et al. (2017) and Wang et al. (2018) utilize external resources, in other words, phrase dictionary and semantic annotations. Li et al. (2018) reinforce the paraphrase generator by a learnt reward function. Although achieving state-of-the-art performances, none of the above work considers the paraphrase patterns at different levels of granularity. Moreover, their models can generate the paraphrase in a neither interpretable nor a fine-grained controllable way. In Iyyer et al. (2018)'s work, the model is trained to produce a paraphrase of the sentence with a given syntax. In this work, we consider automatically learning controllable and interpretable paraphrasing operations from the corpus. This is also the first work to consider scalable unsupervised domain adaptation for neural paraphrase generation.

4.2 Controllable and Interpretable Text Generation

There is extensive attention on controllable neural sequence generation and its interpretation. A line of research is based on variational auto-encoder (VAE), which captures the implicit (Gupta et al., 2017; Li et al., 2017) or explicit information (Hu et al., 2017; Liao et al., 2018) via latent representations. Another approach is to integrate probabilistic graphical model, e.g., hidden semi-Markov model (HSMM) into neural network (Wiseman et al., 2018; Dai et al., 2016). In these works, neural templates are learned as a sequential composition of segments controlled by the latent states, and be used for language modeling and data-to-text generation. Unfortunately, it is non-trivial to adapt this approach to the Seq2Seq learning framework to extract templates from both the source and the target sequence.

4.3 Domain Adaptation for Seq2Seq Learning

Domain adaptation for neural paraphrase generation is under-explored. To our best knowledge, Su and Yan (2017)'s work is the only one on this topic. They utilize the pre-trained word embedding and include all the words in both domains to vocabulary, which is tough to scale. Meanwhile, we notice that there is a considerable amount of work on domain adaptation for neural machine translation, another classic sequence-to-sequence learning task. However, most of them require parallel data in the target domain (Wang et al., 2017a,b). In this work, we consider unsupervised domain adaptation, which is more challenging, and there are only two works that are applicable. Gulcehre et al. (2015) use the language model trained in the target domain to guide the beam search. Domhan and Hieber (2017) optimize two stacked decoders jointly by learning language model in the target domain and learning to translate in the source domain. In this work, we utilize the similarity of sentence templates in the source and target domains. Thanks to the decomposition of multi-grained paraphrasing patterns, DNPG can fast adapt to a new domain without any parallel data.

5 Conclusion

In this paper, we have proposed a neural paraphrase generation model, which is equipped with a decomposition mechanism. We construct such mechanisms by latent variables associated with each word, and a couple of Transformer models with various inductive biases to focus on paraphrase patterns at different levels of granularity. We further propose a fast and incremental method for unsupervised domain adaptation. The quantitative experiment results show that our model has competitive in-domain performance compared to the state-of-the-art models, and outperforms significantly upon other baselines in domain adaptation. The qualitative experiments demonstrate that the generation of our model is interpretable and controllable. In the future, we plan to investigate more efficient methods of unsupervised domain adaptation with decomposition mechanism on other NLP tasks.

References

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Thirty-First AAAI Conference on Artificial Intelligence*.

Hanjun Dai, Bo Dai, Yan-Ming Zhang, Shuang Li, and Le Song. 2016. Recurrent hidden semi-markov model. In *International Conference on Learning Representations*.

Tobias Domhan and Felix Hieber. 2017. Using target-side monolingual data for neural machine translation through multi-task learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1500–1505.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 1631–1640.

Caglar Gulcehre, Orhan Firat, Kelvin Xu, Kyunghyun Cho, Loic Barrault, Huei-Chi Lin, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2015. On using monolingual corpora in neural machine translation. *arXiv preprint arXiv:1503.03535*.

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

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. *arXiv preprint arXiv:1703.00955*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. *arXiv preprint arXiv:1804.06059*.

Eric Jang, Shixiang Gu, and Ben Poole. 2016. Categorical reparameterization with gumbel-softmax. *arXiv preprint arXiv:1611.01144*.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Yann LeCun, Léon Bottou, Yoshua Bengio, Patrick Haffner, et al. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. *arXiv preprint arXiv:1708.00625*.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3865–3878.

Yi Liao, Lidong Bing, Piji Li, Shuming Shi, Wai Lam, and Tong Zhang. 2018. Quase: Sequence editing under quantifiable guidance. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3855–3864.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*.

Shuming Ma, Xu Sun, Wei Li, Sujian Li, Wenjie Li, and Xuancheng Ren. 2018. Query and output: Generating words by querying distributed word representations for paraphrase generation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, volume 1, pages 196–206.

Franz Josef Och and Hermann Ney. 2003. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29(1):19–51.

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*, pages 311–318. Association for Computational Linguistics.

Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. In *International Conference on Learning Representations*.

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*.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

Yu Su and Xifeng Yan. 2017. Cross-domain semantic parsing via paraphrasing. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1235–1246.

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*, pages 38–42. Association for Computational Linguistics.

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

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*, pages 5998–6008.

Rui Wang, Andrew Finch, Masao Utiyama, and Eiichiro Sumita. 2017a. Sentence embedding for neural machine translation domain adaptation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 560–566.

Rui Wang, Masao Utiyama, Lemao Liu, Kehai Chen, and Eiichiro Sumita. 2017b. Instance weighting for neural machine translation domain adaptation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1482–1488.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2018. A task in a suit and a tie: paraphrase generation with semantic augmentation. *arXiv preprint arXiv:1811.00119*.

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

Sam Wiseman, Stuart Shieber, and Alexander Rush. 2018. Learning neural templates for text generation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3174–3187.

A Algorithm for extracting templates

Algorithm 1 ExtractSentParaPattern

INPUT: X, Y, Z^{x}, Z^{y}, α', V
OUTPUT: $\bar{X}$, $\bar{Y}$

procedure EXTRACT $\bar{X}$
  $L \leftarrow |X|$;
  $\bar{X} \leftarrow [\,]$;
  $c \leftarrow 1$;
  $p \leftarrow [\,]$;
  for $l := 1$ to L **do**
    if $z_l^{\mathrm{x}} = 0$ **then**
      if $l = 1$ **or** $\bar{X}_{l-1} \notin V$ **then**
        $\bar{X}$.add(V_c);
        p.add($[\,]$);
        $c \leftarrow c + 1$;
      else
        $p[c]$.add(l);
    else
      $\bar{X}$.add(X_l);
procedure EXTRACT $\bar{Y}$
  $T \leftarrow |Y|$;
  $\bar{Y} \leftarrow [\,]$;
  for $t := 1$ to T **do**
    if $z_t^{\mathrm{y}} = 0$ **then**
      $c \leftarrow \arg\max_c \frac{1}{|p[c]|} \sum_{l=1}^{|p[c]|} \alpha'_{p[c][l],t}$;
      if $t = 1$ **or** $\bar{Y}_{t-1} \neq V_c$ **then**
        $\bar{Y}$.add(V_c);
    else
      $\bar{Y}$.add(Y_t);
End

Table 10: Examples of the generated paraphrases and extracted patterns at each granularity level by DNPG.

Input Sentence	Extracted Source Templates	Generated Sentential Paraphrase	Generated Paraphrase	Generated by MTL+Copy
is there any verify angel investor on quora?	is there any $x on $y	how many $x on $y	how many verify angel investor on quora?	is there any verify on quora?
how much salary do iit professor get?	how much salary do $x get	how much money do $x make	how much money do iit professor make?	how much do professor UNK?
which is the best mobile below 15000?	which is the $x	the $x	the best mobile below 15000 ?	what mobile 15000?
how many time should i have bath?	how many $x	number of $x	number of time should i have bath?	how do you have bath?
who is the best hollywood actor?	who is the $x	what is $x name	what is the best hollywood actor name?	who is the best actor?
how do you change a key ignition in a 1988 chevy celebrity?	how do you $x	what is the best way to $x	what is the best way to change a key ignition in a 1988 chevy celebrity?	how do you change a 1988 in a 1988 chevy?
why do company issue bonus share ?	why do $x	what is the purpose of the $x	what is the purpose of the company issue bonus share?	how do company issue bonus share?
under which condition do the hiv virus survive?	under which condition do the $x	which condition is best for the $x	which condition is best for the hiv virus survive?	which do the hiv virus survive?
use of monggo seed ?	$x of $y	what is the $x of $y	what is the use of monggo seed?	?
how do you eat potato salad?	how do you $x	is there any way to $x	is there any way to eat potato salad?	how do you eat potato salad?
who is the most important person in yours life?	who is the $x in $y	who is $y 's $x	who is yours life 's most important person?	what is the most important person in yours life?
what is the easiest business to start?	what is $x	what is $x in the world	what is the easiest business to start in the world?	what is business?

B Evaluation Guide

Please evaluate the paraphrase with respect to three criterions: readability, accuracy, and diversity, which are arranged by importance. Specifically, the criterions of paraphrase quality from bad to good are listed in detailed as following:

1. Non-readable. The generated paraphrase does not make sense and is not human-generated text. Please note that readable is not equivalent to grammatical correct. That is, considered there are non-English speaker, a readable paraphrase can have grammar mistakes.

2. Readable but is not accurate. The answer to the paraphrased question is not helpful to the owner of the original question. For instance, *how can i study c++* $\rightarrow$ *what be c++*. Here are some examples of accurate paraphrase:

   (a) *how can i learn c++* $\rightarrow$ *what be the best way to learn c++*

   (b) *can i learn c++ in a easy way* $\rightarrow$ *be learn c++ hard*

   (c) *do you have some suggestion of well design app* $\rightarrow$ *what be some well design app name*

   (d) *be study hard* $\rightarrow$ *how study hard*

3. Accurate but with trivial paraphrasing. Just remove or add some stop words. For instance, *why can trump win the president election* $\rightarrow$ *why can trump win president election*

4. Novel paraphrasing. More or loss, there is information loss of a non-trivial paraphrase. Thus, again, determine whether the paraphrase is equivalent to the original question from the perspective of question owner. Furthermore, it is not necessary for a non-trivial paraphrase contains rare paraphrasing pattern. For instance, maybe there is lot of paraphrase with the pattern *what be $name* $\rightarrow$ *some interesting facts about $name*. But it can still be considered as non-trivial paraphrase.

There are some other things to be noted:

1. There maybe special token, that is, [UNK] in the generated paraphrase. A generated paraphrase with [UNK] should generally have higher rank.

2. The same paraphrase should have same ranking. Otherwise, please try your best to distinguish the quality of paraphrase.

3. Please do Google search first when you see some strange word or phrase for better evaluation.

4. Please note that all the words are stemmed and lower case. Just assume all the words are in their right form. For instance, *what be you suggestion of some english movie* is equivalent to *What are your suggestions of some English movies*.