Learning Structural Information for Syntax-Controlled Paraphrase Generation

Erguang Yang[1] , **Chenglin Bai**[1] , **Deyi Xiong**[2], **Yujie Zhang**[1]*, **Yao Meng**[3] , **Jinan Xu**[1] , **Yufeng Chen**[1]
[1]School of Computer and Information Technology, Beijing Jiaotong University, Beijing, China
[2]College of Intelligence and Computing, Tianjin University, Tianjin, China
[3]Lenovo Research AI Lab, Beijing, China
{egyang,chenglin.bai,yjzhang,jaxu,chenyf}@bjtu.edu.cn,
dyxiong@tju.edu.cn, yaomeng1@lenovo.com

Abstract

Syntax-controlled paraphrase generation aims to produce paraphrase conform to given syntactic patterns. To address this task, recent works have started to use parse trees (or syntactic templates) to guide generation. A constituency parse tree contains abundant structural information, such as parent-child relation, sibling relation, and the alignment relation between words and nodes. Previous works have only utilized parent-child and alignment relations, which may affect the generation quality. To address this limitation, we propose a **S**tructural **I**nformation-augmented **S**yntax-**C**ontrolled **P**araphrasing (SI-SCP) model. Particularly, we design a syntax encoder based on tree-transformer to capture parent-child and sibling relations. To model the alignment relation between words and nodes, we propose an attention regularization objective, which makes the decoder accurately select corresponding syntax nodes to guide the generation of words. Experiments show that SI-SCP achieves state-of-the-art performances in terms of semantic and syntactic quality on two popular benchmark datasets. Additionally, we propose a **S**yntactic **T**emplate **R**etriever (STR) to retrieve compatible syntactic structures. We validate that STR is capable of retrieving compatible syntactic structures. We further demonstrate the effectiveness of SI-SCP to generate diverse paraphrases with retrieved syntactic structures.

1 Introduction

Paraphrases are texts or passages conveying the same meaning but with different surface realization. Paraphrase generation (PG) is a key technology of automatically generating a restatement for a given text, which has the potential use in many downstream tasks, such as question answering (Gan and Ng, 2019), machine translation (Zhou et al., 2019), and sentence simplification (Zhao et al., 2018). However, a natural sentence can be paraphrased into various surface forms.

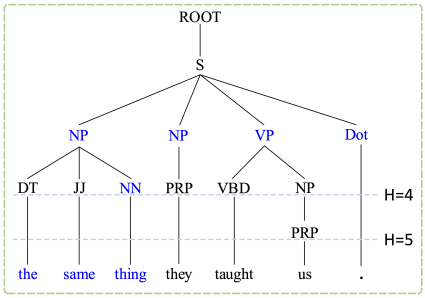


Figure 1: An example of a constituency tree structure. Parent-child relation: NP→NN, sibling relation: NP NP VP Dot, alignment relation between nodes and words: (NN thing), (NP the same thing).

To obtain diverse paraphrases, controllable paraphrase generation (CPG) with specified styles has recently attracted growing interests e.g., satisfying particular sentiment (Hu et al., 2017; John et al., 2019; Dai et al., 2019; Lee et al., 2021) or syntactic structure (Iyyer et al., 2018; Chen et al., 2019; Liu et al., 2020; Kumar et al., 2020; Li et al., 2020; Yang et al., 2021). As CPG can produce diverse paraphrases by exposing syntactic control, it can be used in a wide range of application scenarios, such as dialogue generation (Niu and Bansal, 2018), data augmentation (Iyyer et al., 2018; Yang et al., 2021; Sun et al., 2021), and diverse question generation (Yu and Jiang, 2021), etc.

Generally, syntax-controlled paraphrase generation needs to tackle two major challenges. *The first challenge is how syntax-controlled generating is achieved?* For this challenge, Iyyer et al. (2018) use two encoders to encode input sentences and linearized parse trees to produce paraphrases. A constituency parse tree contains abundant structural information, such as parent-child relation, sibling relation, and the alignment relation between words and nodes, as shown in Figure 1. Linearizing parse trees, typically, result in loss of structural informa-

*Corresponding Author.

Findings of the Association for Computational Linguistics: NAACL 2022, pages 2079 - 2090

tion. Kumar et al. (2020) encode parse trees in a top-down manner, and then a queue-based decoding mechanism is proposed to incorporate syntax information. Li et al. (2020) employ a path attention mechanism to capture the tree structure of the syntax. However, these methods still have some limitations. The top-down encoding manner and path attention only consider parent-child relation. Additionally, due to the complexity of the queue-based decoding mechanism (Kumar et al., 2020), the predicted `pop` sequences cannot be guaranteed to be perfect, which may cause error propagation.

The second challenge is how compatible syntactic structures can be retrieved to guide generation in practice? Previous works (Iyyer et al., 2018; Li et al., 2020) use common syntactic templates that appear most frequently from the corpus, which means that these works generate syntactically diverse paraphrases using a fixed set of syntactic structures for all input sentences. The diversity in syntax is hence limited. Moreover, not all sentences can be paraphrased into the same set of syntactic structures.

In order to address these problems, we propose a **S**tructural **I**nformation-augmented **S**yntax-**C**ontrolled **P**araphrasing (SI-SCP) model based on attention network. Particularly, we design a tree transformer to capture parent-child and sibling relation. To learn the alignment relation between words and nodes, we propose an attention regularization objective. The basic motivation is that a syntactic template contains several syntax nodes which guide the generation of different words, respectively. As shown in Figure 1, the `NP` node guides the generation of noun phrase. Learning alignment relation makes the decoder accurately select corresponding syntax nodes to guide the generation of words. Additionally, to enhance diversity in syntax-controlled generation, we propose a **S**yntactic **T**emplate **R**etriever (STR) to retrieve compatible syntactic structures for any input sentence.

We evaluate our model on two popular benchmark datasets. Experiment results show that SI-SCP achieves the state-of-the-art performence in syntactic and semantic quality. The visualization results show the attention regularization makes the decoder accurately attend to corresponding syntactic nodes during decoding. Human evaluation also demonstrates that our method is able to generate semantically and syntactically better sentences than previous methods. We further show that STR can retrieve more compatible structures compared with the common syntactic templates method. SI-SCP can generate more syntactically diverse paraphrases with retrieved syntactic structures.

In summary, the major contributions of this paper are as follows:

- We build a novel syntax-controlled paraphrasing model that contains a tree-transformer and an attention regularization objective.
- To retrieve compatible syntactic structures in practice, we propose a syntactic template retriever.
- Experiments show that our SI-SCP achieves new state-of-the-art results in both semantic and syntactic evaluation on two popular benchmark datasets. We further demonstrate that STR is capable of retrieving compatible syntactic templates. The SI-SCP can produce more syntactically diverse paraphrases with retrieved syntactic structures.

2 Related Work

We focus primarily on the task of syntactically controlled paraphrase generation, which has recently attracted increasing attention (Iyyer et al., 2018; Chen et al., 2019; Liu et al., 2020; Kumar et al., 2020; Li et al., 2020). According to the control element, previous works can be divided into two categories. The first strand of research uses sentential exemplar as syntactic control. Chen et al. (2019) and Liu et al. (2020) design a latent variable to learn syntax by encoding the exemplar itself, and then use a syntactic latent variable to guide the generation of paraphrases. However, a latent based approach might not offer enough explicit syntactic information as guaranteed by actual constituency parse trees (Kumar et al., 2020). The second strand of research takes a parse tree as a syntactic input, controlling the syntax of generated text with the structure specified by the parse tree. Iyyer et al. (2018) use two encoders to encode input sentences and linearized parse trees to produce paraphrases. Due to the linearization of syntactic tree, a lot of structural information is generally lost. To capture the tree structure of the parse tree, Kumar et al. (2020) encode parse trees in a top-down manner, and then a queue-based decoding mechanism is proposed to incorporate syntactic information. Li

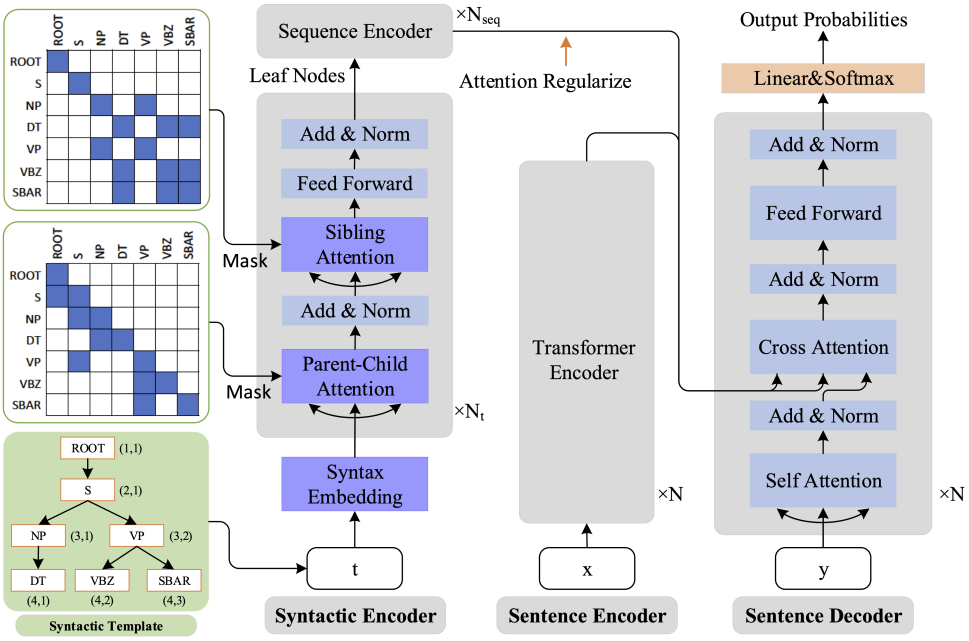


Figure 2: The architecture of syntax-controlled paraphrasing model. The sequence encoder is built with the transformer network, which is used to encode leaf nodes in the syntactic template. Please refer Section 3.3 for details.

et al. (2020) design a syntax encoder based on a path attention mechanism. However, these methods only consider parent-child relation in parse trees.

Differing from previous approaches, we propose a novel tree-transformer to model parent-child and sibling relation for better capturing the tree structure of the parse tree. To learn the alignment relation between words and nodes, we also propose an attention regularization objective, which makes the decoder accurately select corresponding syntax nodes to guide the generation of words. Additionally, we propose a syntactic template retriever that can help to retrieve compatible syntactic structures for any input sentences.

3 Syntax-Controlled Paraphrasing

3.1 Problem Formalization

We formulate the problem of syntax-controlled paraphrase generation as follows. Given a source sentence $\boldsymbol{x}$ and a syntactic template $\boldsymbol{t}$, the syntax-controlled text generation aims to generate target sentence $\boldsymbol{y}$ which conveys the meaning of $\boldsymbol{x}$ and conform to the syntactic structure of $\boldsymbol{t}$.

The syntactic template $\boldsymbol{t}$ is a partial constituency parse tree that provides a general syntax skeleton. Due to different levels of syntax trees contain different information, to compare fairly, we use the top-4 layers (in this work) of the full parse tree of $\boldsymbol{y}$ as the syntactic template for all baselines. Of course, our model can also use syntactic templates from other layers.

As shown in Figure 2, our model contains a sentence encoder and a syntactic encoder, which encodes the source sentence $\boldsymbol{x}$ and the template $\boldsymbol{t}$ separately. We deploy a target sentence decoder to generate the final text. The details of the proposed approach will be presented below.

3.2 Sentence Encoder

The sentence encoder contains N stacked transformer blocks. The hidden states of the sentence encoder are calculated by:

$$\boldsymbol{h}_x = \texttt{Transformer}(\boldsymbol{x}) \quad (1)$$

where $\boldsymbol{h}_x$ is final hidden state of source sentence $\boldsymbol{x}$.

3.3 Syntactic Encoder

This encoder provides the necessary syntactic signal for the generation of paraphrases. A parse tree contains rich parent-child and sibling relations. To capture these information, we propose a tree-transformer with a syntax embedding layer, parent-child attention, and sibling attention modules.

Syntax Embedding Layer Given a syntactic template $\boldsymbol{t}$, we use (node, level, index) format

sequences to represent it. Formally, let $\boldsymbol{t} = \{\boldsymbol{t}_1, \boldsymbol{t}_2, ..., \boldsymbol{t}_n\}$, where $\boldsymbol{t}_i = (p_i, l_i, s_i)$, p_i is a parse tree node, l_i is its level, s_i is the index of the node at a specific level. For example, the syntactic template in Figure 2 is represented by {(ROOT, 1, 1), (S, 2, 1), (NP, 3, 1), (DT, 4, 1), (VP, 3, 2), (VBZ 4, 2), (SBAR, 4, 3)}.

At the embedding layer, node tokens, level tokens, and index tokens are embedded respectively and then added together to produce the syntax embedding at position i:

$$\texttt{Emb}(\boldsymbol{t}_i) = \texttt{Emb}(p_i) + \texttt{Emb}(l_i) + \texttt{Emb}(s_i) \quad (2)$$

Parent-Child and Sibling Attention Then we calculate the hidden state of each node in a tree-structure manner. Specifically, we introduce an adjacency matrix to guide the calculation of the self-attention module:

$$\begin{aligned} \boldsymbol{A} &= \text{Softmax}((\boldsymbol{Q}\boldsymbol{K}^T/\sqrt{d}) \odot \boldsymbol{M}) \\ \boldsymbol{h}_t &= \boldsymbol{A} \cdot \boldsymbol{V} \end{aligned} \quad (3)$$

where $\boldsymbol{Q}$ is a query matrix consisting of query vectors with dimension d, $\boldsymbol{K}$ is a key matrix consisting of key vectors with dimension d, $\boldsymbol{V}$ is a value matrix consisting of key vectors with dimension d. $\boldsymbol{M}$ is an adjacency matrix obtained from the syntactic template. $\odot$ denotes the element-wise product. For example, Figure 2 shows examples of the parent-child adjacency matrix and sibling adjacency matrix. We encode parent-child and sibling information by parent-child and sibling attention operations. We stack multiple tree transformer blocks to make the information of the ROOT node flow to leaf nodes.

After obtaining hidden states of all nodes, we use another encoder to encode all the leaf nodes in a sequential way:

$$\boldsymbol{h}^{seq}_{leaf} = \texttt{Transformer}(\boldsymbol{h}_{leaf}) \quad (4)$$

The reason we use this sequential encode is that parent-child and sibling attention modules would create a mismatch between the encoding and decoding process. Thus, it may be beneficial to introduce sequence information. We also demonstrate the effectiveness of the sequential encoder with experiments in Section 5.2.4.

3.4 Decoder

Based on the input sentence hidden states $\boldsymbol{h}_x$ and syntactic template hidden states $\boldsymbol{h}^{seq}_{leaf}$, the target transformer decoder uses two cross-attention modules to jointly exploit the input sentence and syntactic template information to generate the target text $\boldsymbol{y}$. Cross attentions are calculated as follows:

$$\begin{aligned} \boldsymbol{h}_{y,t}, \boldsymbol{A}_t &= \text{Attn}(\boldsymbol{h}_y, \boldsymbol{h}^{seq}_{leaf}, \boldsymbol{h}^{seq}_{leaf}) \\ \boldsymbol{h}_{y,t,x}, \boldsymbol{A}_x &= \text{Attn}(\boldsymbol{h}_{y,t}, \boldsymbol{h}_x, \boldsymbol{h}_x) \end{aligned} \quad (5)$$

where Attn is the multi-head attention module in transformer. $\boldsymbol{h}_y$ is hidden states of decoder, $\boldsymbol{A}_t$ and $\boldsymbol{A}_x$ are attention scores with the template and the input sentence, respectively. Through the feed-forward network sub-layer, $\boldsymbol{h}_{y,t,x}$ will be used as the input of the next layer or used to predict the next word.

Syntax Attention Regularization The attention weight $\boldsymbol{A}_t$ is essential for accurate syntactic control. We propose an regularization method to guide the learning of $\boldsymbol{A}_t$ using the alignment of nodes and words in a parse tree. Specifically, we propose a attention regularization loss as follows:

$$\mathcal{L}_{ar} = \text{MSE}(\boldsymbol{A}_t, \hat{\boldsymbol{A}}_t) \quad (6)$$

where $\hat{\boldsymbol{A}}_t$ are oracle attention weights obtained from the alignment of nodes and words in a parse tree. MSE denotes the Mean Square Error loss function. By doing so, we can make learned $\boldsymbol{A}_t$ close to oracle attention distribution. We also introduce label smoothing (0.25) to reduce errors caused by parsing.

Because multi-layer transformer and multi-head attention mechanism produce multiple syntax attention matrices, we simply make each generated attention matrix close to the oracle attention. We leave alternatives to this simple method to our future work.

3.5 Training

To train the above model, we optimize the following objective function:

$$\mathcal{L} = \lambda_1 \mathcal{L}_{ce} + \lambda_2 \mathcal{L}_{ar} \quad (7)$$

where $\mathcal{L}_{ce} = -\sum_{t=1}^{N} \log p(y_t | \boldsymbol{x}, \boldsymbol{t}, y_{1:t-1})$ is the cross-entropy loss for ground-true y, λ_* are balancing hyper-parameters.

4 Syntactic Template Retriever

A sentence cannot be converted to any syntactic structures. Incompatible syntax will lead to imperfect paraphrase conversion and nonsensical

sentences. In this work, we propose a **S**yntactic **T**emplate **R**etriever (STR) to find compatible syntax for a sentence in practice. The basic assumption is that syntactic templates expressing the same meaning are relatively close in semantic space. Specifically, given an input sentence and its parse tree, the retriever retrieves compatible syntactic templates based on similarity from a pre-collected template library.

4.1 Encoder

We use a template encoder to map any template $\boldsymbol{t}$ into a vector space. Meanwhile, for the input sentence $\boldsymbol{x}$ and its parse tree $\boldsymbol{p}_x$ (i.e., the query $\boldsymbol{q}$), we use a sentence encoder and parse tree encoder to map them to the same vector space, and then retrieves k syntactic templates whose vectors are closest to the query vector. We define the similarity between the query and the syntactic template using the dot product of their vectors:

$$\begin{aligned} \text{sim}(\boldsymbol{q}, \boldsymbol{t}) &= \boldsymbol{v}_q \cdot \boldsymbol{v}_t^T \\ \boldsymbol{v}_t &= \text{Enc}_t(\boldsymbol{t}) \\ \boldsymbol{v}_q &= \boldsymbol{W} \cdot [\text{Enc}_x(\boldsymbol{x}); \text{Enc}_p(\boldsymbol{p}_x)] \end{aligned} \tag{8}$$

where Enc_* denote three different encoders, $\boldsymbol{W}$ is a linear layer, ';' denotes concatenation operation.

The three encoders are built with the transformer encoder. We use linearized parse tree and syntactic template as input [1], prepend [CLS] token to each input, and then take the representation of the prepended token as the output of each encoder in a similar way to BERT (Devlin et al., 2019).

4.2 Training

The goal is to create a vector space such that relevant pairs of queries and templates will have a smaller distance (i.e., higher similarity) than the irrelevant pairs. We use paraphrase pairs to build training data, where positive templates are from reference sentences.

Let $(\boldsymbol{q}_i, \boldsymbol{t}_i^+, \boldsymbol{t}_{i,1}^-, ..., \boldsymbol{t}_{i,n}^-)$ be a training instance that consists of one relevant (positive) template $\boldsymbol{t}_i^+$, along with n irrelevant (negative) templates $\boldsymbol{t}_{i,j}^-$. To train the retriever, we optimize the negative log likelihood of the positive template:

$$\begin{aligned} &\mathcal{L}(\boldsymbol{q}_i, \boldsymbol{t}_i^+, \boldsymbol{t}_{i,1}^-, ..., \boldsymbol{t}_{i,n}^-) \\ &= -\log \frac{e^{sim(\boldsymbol{q}_i, \boldsymbol{t}_i^+)}}{e^{sim(\boldsymbol{q}_i, \boldsymbol{t}_i^+)} + \sum_{j=1}^{n} e^{sim(\boldsymbol{q}_i, \boldsymbol{t}_{i,j}^-)}} \end{aligned} \tag{9}$$

At the training stage, we use the trick of in-batch negatives to learn the retriever.

[1] We compared linearized and structured approaches and both have comparable performance, the main reason may be that controlled paraphrase generation requires learning relation between text and syntax (words and nodes), while the retriever learns relation between syntax. Based on the principle of simplicity, we used the linearized approach.

Model	BLEU↑	R-1 / R-2 / R-L↑	MTR↑	TED↓
ParaNMT-Small				
Source-as-Output	18.5	50.6 / 23.2 / 23.1	47.6	11.9
SCPN (2018)	21.2	55.1 / 31.3 / 57.4	33.0	6.3
SGCP (2020)	7.9	34.7 / 13.7 / 36.9	17.9	12.5
GuiG (2020)	26.3	60.7 / 37.1 / 62.5	38.0	6.4
SynTrans	19.3	54.0 / 28.7 / 55.8	32.1	8.0
SI-SCP	**27.8**	**62.8 / 39.5 / 64.4**	**39.9**	**5.7**
- SeqEnc	27.4	61.4 / 37.4 / 63.2	39.1	5.8
- AttnRegu	26.8	61.6 / 38.9 / 63.7	38.5	6.0
- SibAttn	26.1	61.0 / 37.3 / 62.9	38.0	6.1
QQP-Pos				
Source-as-Output	17.2	51.9 / 26.2 / 52.9	31.0	16.2
SCPN (2018)	28.1	59.1 / 36.5 / 62.1	33.1	8.3
SGCP (2020)	9.2	35.8 / 16.0 / 40.2	17.5	12.3
SynTrans	24.8	57.2 / 32.9 / 59.4	33.3	10.8
SI-SCP	**53.5**	**77.3 / 61.0 / 78.8**	**54.5**	**5.2**
- SeqEnc	53.1	76.9 / 60.5 / 78.6	54.4	5.3
- AttnRegu	31.4	61.6 / 38.5 / 63.5	38.0	9.6
- SibAttn	53.0	77.0 / 60.5 / 78.7	54.1	5.2

Table 1: Evaluation results on the ParaNMT-small and QQP-Pos datasets. All scores are reported as the mean over three runs. R-1: ROUGE-1. R-2: ROUGE-2. R-L: ROUGE-L. MTR: METEOR. '-SeqEnc': no sequential encoder used to encode leaf nodes, '-AttnReg': no attention regularization, '-SibAttn': no sibling attention module.

5 Experiments

In this section, we conducted experiments to answer the following questions:

- How does our model compare against previous models?
- Can our model produce diverse paraphrases with retrieved templates?

5.1 Controlled Paraphrase Generation

Implementation details are presented in Appendix A due to limited space.

5.2 Datasets

Following previous work (Kumar et al., 2020), we used ParaNMT-Small and QQP-Pos datasets to

evaluate model performance for controlled paraphrase generation. **ParaNMT-Small** (Chen et al., 2019) contains 500k paraphrase pairs for training, 500 and 800 manually labeled paraphrase pairs for development set and test set. The ParaNMT-small is a subset of the original ParaNMT-50M dataset (Wieting and Gimpel, 2018) which is constructed automatically through back-translation of original English sentences. **QQP-Pos** (Kumar et al., 2020) is selected from Quora Question Pairs (QQP) dataset. It contains about 140K training pairs and 3K/3K data pairs for testing/validation.

5.2.1 Baselines[2]

- Source as output: Simply output the source sentence as output.

- SCPN (Iyyer et al., 2018) & GuiG (Li et al., 2020): They adopt a two-stage generation process. Iyyer et al. (2018) use two Bi-LSTM (Hochreiter and Schmidhuber, 1997) encoders to encode input sentences and linearized parse trees respectively. An LSTM decoder with attention mechanism pays attention to both semantic and syntactic hidden states to generate paraphrases. Li et al. (2020) use transformer network and propose a syntactic encoder with a path attention mechanism. GuiG only provided trained model on the ParaNMT-Small dataset.

- SGCP (Kumar et al., 2020): SGCP encodes syntactic templates with GRU network in a top-down manner. We directly used released SGCP model[3] and top-4 layers of the full parse tree of the reference paraphrases as syntactic templates.

- SynTrans: The framework is similar to SCPN, and we replace LSTM with Transformer (Vaswani et al., 2017). In this experiment, we directly used linearized syntactic templates to guide generation.

5.2.2 Automatic Evaluation

1. Semantic Metrics: BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), and METEOR (Banerjee and Lavie, 2005) scores between generated sentences and reference paraphrases in the test set were used as semantic metrics.

2. Syntactic Metrics: Following previous works (Chen et al., 2019; Kumar et al., 2020; Li et al., 2020), we used the tree edit distance (TED) against the parse tree of the reference.

Model	Semantic	Syntactic
SCPN	3.16	3.73
GuiG	3.67	3.77
SynTrans	3.05	3.51
SI-SCP	3.81	4.55

Table 2: Human evaluation results.

5.2.3 Human Evaluation

We conducted the human evaluation on 100 randomly selected instances from the test set of ParaNMT-Small in a blind fashion. Three annotators evaluated generated paraphrases in terms of semantic and syntactic similarity (generations vs references); each aspect was scored from 1 to 5.

5.2.4 Results

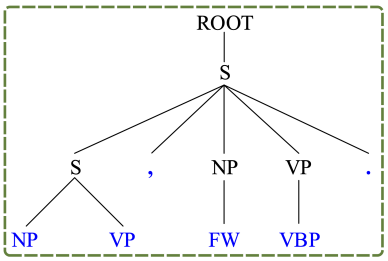
As can be observed in Table 1, firstly, SGCP gets the lowest results, we observe that SGCP often produces sentences that end abruptly, thereby harming syntax and semantics[4]. GuiG achieves better performance than SCPN model among two-stage approach. This is because GuiG adopts the more advanced transformer network and a path-attention mechanism which can capture partial structural information. For SynTrans which uses linerized syntactic templates directly, it obtains relatively lower performance. Conversely, SI-SCP achieves significant improvements in both semantic and syntactic metrics, which means that modeling structural information can improve the quality of generations.

Among different variants of our model (ablation study), we see that removing any of the sequence encoder, attention regularization, and sibling attention module has a negative impact on performance, where attention regularization and sibling attention modules have a more significant impact.

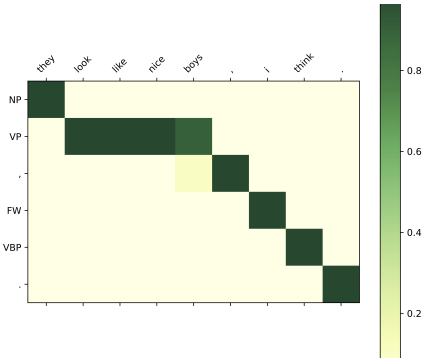
[2]Due to the different control types, we don't compared with the example-based methods (Chen et al., 2019; Liu et al., 2020)

[3]https://github.com/malllabiisc/SGCP

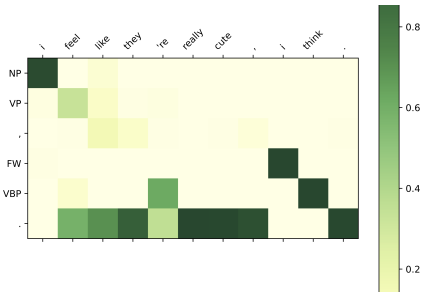
[4]The results of the SGCP baseline reported here are lower than in the original paper. This is because that original paper used very fine-grained syntactic information, which contains more hints about reference paraphrase. Additionally, compared with syntactic templates, fine-grained syntactic structures are more conducive to guiding models to generate natural sentences.



(a) Syntactic template



(b) Syntax Attention with Regularization



(c) Syntax Attention without Regularization

Figure 3: (a) shows the used syntactic template. (b) and (c) show visualization results of syntax attention with or without regularization.

Table 2 shows the results of human evaluation which are consistent with automatic evaluation results. Our model obtains the highest scores in both semantics and syntax, thereby highlighting the efficacy of our method.

Table 6 shows examples of paraphrases generated by different model. We can observe that our SI-SCP can produce better results than baseline models in terms of both semantics and syntax.

5.2.5 Visualization of Syntax Attention

We visualize the syntax attention when using the syntactic template as control in Figure 3. This is an instance from the test dataset: {***Source***: *it looks to me like they are really sweet boys.* ***Reference***: *they look like very nice boys, i think.*}. Figure 3 (a) is the syntactic tempalte used in the example.

We can see that most words can accurately align with corresponding nodes when using attention regularization. Without attention regularization, most words tend to pay attention to the punctuation. These results show that the proposed attention regularization can make the decoder accurately select corresponding nodes to guide the generation of words.

Dataset	Model	Top-1	Top-5	Top-10	Top-20
ParaNMT -Small	CT (2018)	2.5	6.7	11.3	16.7
	STR	**8.6**	**20.5**	**27.3**	**34.4**
QQP -Pos	CT (2018)	9.2	40.2	47.9	58.8
	STR	**19.1**	**48.7**	**61.1**	**71.1**

Table 3: Retrieval accuracy (%). 'CT': common template, 'STR': our syntactic template retriever.

Model	B-1 / B-2 / B-3 / B-4↑	Re(%)↓	Slf-B↓	Valid(%)↑
ParaNMT-Small				
Seq2Seq	55.8 / 39.1 / 28.8 / 21.9	30.0	69.2	56.7
(1) CT & SCPN	48.1 / 31.0 / 20.6 / 14.0	70.7	34.4	15.1
(2) CT & GuiG	51.0 / 34.4 / 23.9 / 17.2	56.5	30.4	33.8
(3) CT & SI-SCP	51.8 / 35.7 / 25.3 / 18.3	64.1	**28.7**	36.8
(4) STR & SCPN	55.1 / 37.7 / 26.2 / 18.6	46.0	32.5	24.5
(5) STR & GuiG	56.2 / 38.6 / 27.2 / 19.5	**39.7**	32.0	55.7
(6) STR & SI-SCP	**57.6 / 40.6 / 29.5 / 21.7**	42.1	32.2	**65.7**
QQP-Pos				
Seq2Seq	53.0 / 39.6 / 30.9 / 24.9	9.4	89.3	28.5
(7) CT & SCPN	60.4 / 43.7 / 32.3 / 24.4	40.2	28.2	52.9
(8) CT & SI-SCP	68.1 / 54.7 / 45.3 / 38.4	25.1	**27.0**	62.2
(9) STR & SCPN	62.7 / 46.7 / 35.6 / 27.6	28.1	28.1	63.5
(10) STR & SI-SCP	**71.8 / 59.4 / 50.5 / 43.9**	**14.4**	31.5	**67.6**

Table 4: Diverse paraphrase generation results. STR&SI-SCP denotes that SI-SCP uses 10 templates retrieved by the STR to generate 10 paraphrases. CT denotes using 10 common templates. B-1: BLEU-1. B-2: BLEU-2. B-3: BLEU-3. B-4: BLEU-4. Re: Rejection. Slf-B: Self-BLEU.

5.3 Diverse Paraphrase Generation

In this section, we further evaluated our model's ability to generate diverse paraphrases. We first examine whether STR can retrieve compatible templates. We generated 10 syntactically different paraphrases for each input sentence using 10 retrieved syntactic templates. Implementation details are presented in Appendix B.

5.3.1 Evaluation Metrics

We evaluated this task with the following metrics:

1. Retrieval Accuracy: We use Top-K retrieval accuracy on the test set, measured as the percentage of Top-K retrieved templates that contain the gold template. The gold template is

Template	Paraphrase
Source	you can choose between a movie or a pottery class.
(S (NP (PRP)) (VP (MD) (VP)) (.))	you can choose from the film or the pottery class.
(S (NP (NP) (CC) (NP)) (VP (MD) (VP)) (.))	the film or the pottery class can be selected.
(S (NP (NP) (,) (CC) (NP)) (VP (MD) (VP)) (.))	the film , or the pottery class can be selected.
(SQ (MD) (NP (PRP)) (VP (VP) (CC) (VP)) (.))	can you pick one of the film or the pottery class?
(S (NP (PRP)) (VP (VBZ) (UCP)) (.))	it 's possible to choose between film or the pottery class.
(S (NP (RB) (DT) (JJ) (NN)) (VP (MD) (VP)) (.))	perhaps a pottery class can choose between the movies and the film.

Table 5: Syntactic paraphrases generated by SI-SCP with retrieved templates. We show several successful and one failed (in blue) generations.

obtained from the reference sentence. This metric is used to evaluate the performance of the template retriever.

2. Semantics: Given 10 generated paraphrases $Y = \{y^1, y^2, ..., y^{10}\}$ for each input sentence in the test set. For 10 paraphrases, the one with the highest BLEU score to the reference sentence is selected as the final generation y^{best}. The BLEU score between (y^{best}, y) is calculated at the corpus level.

3. Rejection Rate: We use Sentence-BERT [5] (Reimers and Gurevych, 2019) to compute paraphrase scores for generated outputs with respect to the input. And then use this score[6] to filter out low-quality paraphrases. The percentage of filtered sentences is taken as a rejection rate.

4. Diversity: We compute BLEU between all pairs (y^i, y^j), then macro-average these values at the corpus-level.

5. Validity (Valid): To measure paraphrase quality, we perform human evaluation on 100 randomly selected paraphrases from the remaining paraphrases. Three annotators evaluate whether the generated sentences are true paraphrases, (the paraphrase is marked with 1, otherwise marked with 0). Then we compute the percentage of paraphrases marked as 1.

5.3.2 Results

As can be observed in Table 3, our STR significantly surpasses CT in retrieval accuracy. These results show that STR is capable of retrieving compatible syntactic templates.

In Table 4, we also show the results of the vanilla Seq2Seq based on transformer, where we use top-K (K=50) sampling to generate 10 paraphrases. Because this method tends to generate repeated sentences, it obtains lower valid scores on the QQP-Pos dataset. We see that syntax-controlled paraphrasing method significantly improve the diversity of generations.

Among different syntax-controlled models, compared with CT, STR significantly improves semantics, rejection rate, and validity metrics (Row 1/2/3/7/8 vs. Row 4/5/6/9/10). These results validate the advantages of the syntactic template retriever from the perspective of practical application. Using the same syntactic templates, GuiG can get a better rejection rate, but SI-SCP obtains better performance in terms of semantics, validity metrics. These results are consistent with automatic evaluation results in Table 1.

5.3.3 Case Study

Table 5 lists some paraphrases generated by SI-SCP with different syntactic templates. More generation results are presented in Appendix C. We see that the generated sentences always conform to the target templates. These examples are well-formed, semantically sensible, and grammatically correct sentences that also preserve semantics of the original sentences. However, our model also produces sentences with semantic deviation, like the failed cases in Table 5, when given template is incompatible with the input sentence.

6 Conclusion

We have presented a Structural Information-augmented Syntax-Controlled Paraphrasing (SI-SCP) model which can directly generate syntactic paraphrases with syntactic templates. Particularly,

[5] We used the paraphrase-distilroberta-base-v1, which is trained on large-scale paraphrase data. Available at: https://public.ukp.informatik.tu-darmstadt.de/reimers/sentence-transformers/v0.2/

[6] Similar to Iyyer et al. (2018), we set minimum paraphrase similarity to 0.7.

we propose a tree-transformer and an attention regularization. The tree transformer can model parent-child and sibling relation of the syntactic template. The attention regularization method makes the decoder accurately select corresponding syntax nodes to guide the generation of words. To retrieve compatible syntactic templates in practice, we further propose a Syntactic Template Retriever (STR). Experiments show that SI-SCP achieves substantial improvements over previous strong baselines. Furthermore, we also validate that STR is capable of retrieving compatible syntactic templates. SI-SCP can produce more syntactically paraphrases with retrieved syntactic templates.

Acknowledgements

The present research was supported by the National Nature Science Foundation of China (No. 61876198, 61976015, 61976016). Deyi Xiong was supported by the Natural Science Foundation of Tianjin (Grant No. 19JCZDJC31400). We would like thank the four anonymous reviewers for their constructive suggestions and insightful comments.

References

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, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In ACL, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In ACL, pages 6065–6075, Florence, Italy. Association for Computational Linguistics.

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. In Proceedings of the 34th International Conference on Machine Learning - Volume 70, ICML'17, page 1587–1596. JMLR.org.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In NAACL, pages 1875–1885. Association for Computational Linguistics.

Vineet John, Lili Mou, Hareesh Bahuleyan, and Olga Vechtomova. 2019. Disentangled representation learning for non-parallel text style transfer. In Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics, pages 424–434, Florence, Italy. Association for Computational Linguistics.

Diederik Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. Computer Science.

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha Talukdar. 2020. Syntax-guided controlled generation of paraphrases. TACL, 8:329–345.

Dongkyu Lee, Zhiliang Tian, Lanqing Xue, and Nevin L. Zhang. 2021. Enhancing content preservation in text style transfer using reverse attention and conditional layer normalization. In Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers), pages 93–102, Online. Association for Computational Linguistics.

Yinghao Li, Rui Feng, Isaac Rehg, and Chao Zhang. 2020. Transformer-based neural text generation with syntactic guidance.

Zichao Li, Xin Jiang, Lifeng Shang, and Hang Li. 2018. Paraphrase generation with deep reinforcement learning. In EMNLP, pages 3865–3878, Brussels, Belgium. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In Text Summarization Branches Out, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Mingtong Liu, Erguang Yang, Deyi Xiong, Yujie Zhang, Chen Sheng, Changjian Hu, Jinan Xu, and Yufeng Chen. 2020. Exploring bilingual parallel corpora

for syntactically controllable paraphrase generation. In IJCAI-20, pages 3955–3961. International Joint Conferences on Artificial Intelligence Organization. Main track.

Christopher Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In ACL, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Tong Niu and Mohit Bansal. 2018. Polite Dialogue Generation Without Parallel Data. Transactions of the Association for Computational Linguistics, 6:373–389.

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, ACL '02, page 311–318, USA. Association for Computational Linguistics.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In EMNLP, Hong Kong, China. Association for Computational Linguistics.

Jiao Sun, Xuezhe Ma, and Nanyun Peng. 2021. AESOP: Paraphrase generation with adaptive syntactic control. In EMNLP, pages 5176–5189, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

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.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In ACL, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

Erguang Yang, Mingtong Liu, Deyi Xiong, Yujie Zhang, Yao Meng, Changjian Hu, Jinan Xu, and Yufeng Chen. 2021. Syntactically-informed unsupervised paraphrasing with non-parallel data. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 2594–2604, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Xiaojing Yu and Anxiao Jiang. 2021. Expanding, retrieving and infilling: Diversifying cross-domain question generation with flexible templates. In EACL, pages 3202–3212, Online. Association for Computational Linguistics.

Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018. Integrating transformer and paraphrase rules for sentence simplification. In EMNLP, pages 3164–3173, Brussels, Belgium. Association for Computational Linguistics.

Zhong Zhou, Matthias Sperber, and Alexander Waibel. 2019. Paraphrases as foreign languages in multilingual neural machine translation. In ACL, pages 113–122, Florence, Italy. Association for Computational Linguistics.

A Controlled Paraphrase Generation

A.1 Implementation Details

We parsed all sentences in the training set, reference sentences in the validation and test set using Stanford CoreNLP (Manning et al., 2014). We used the scheduled Adam optimizer (Kingma and Ba, 2014) for optimization, and the learning rate was set to 2.0 for all experiments. We set hidden state size to 256 (i.e., d), filter size to 1024, head number to 4. The number of layers of the sentence encoder, sentence decoder, tree transformer and sequence encoder were set to 4, 4, 3, and 2, respectively. The batch size was set to 128. λ_1 was set to 5.0 while λ_2 1.0. We used BPE tokens pre-trained with $30,000$ iterations. All hyperparameter tuning was based on the BLEU score on the development set.

For the SynTrans baseline model, we set the number of syntactic encoder layers to 5 for fair comparison.

B Diverse Paraphrase Generation

B.1 Implementation Details

For the syntactic template retriever, we used 300 as hidden size, 512 as filter size and 4 heads in multi-head attention. The number of layers of sentence, syntax, and template encoders were all set to 4. The batch size was set to 512. We used the scheduled Adam optimizer (Kingma and Ba, 2014) for optimization, and the learning rate was set to 0.1. The word embedding layer was initialized by the publicly available GloVe (Pennington et al., 2014) 300-dimensional embeddings.[7]

C Generated Paraphrase Examples

Table 6 shows several paraphrases generated by each model. Table 7 lists some paraphrases generated by SI-SCP with different syntactic templates. Our model can generate more syntactically diverse paraphrases with retrieved syntactic templates.

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

Source	it looks to me like they are really sweet boys.
Reference	they look like very nice boys , i think.
SGCP	i have no idea.
SCPN	you know , they 're a lot of cute guys.
SynTrans	i think they 're really cute boys,
GuiG	i do n't like nice guys , i think.
SI-SCP	they look like nice boys , i think.
Source	have you seen this man since your brother was killed?
Reference	after the murder of your brother , did you see this man?
SGCP	since your brother , did you see your brother?
SCPN	so , did you see the guy since your brother was murdered?
SynTrans	after your brother , did you see that you killed him?
GuiG	since your brother , did you see the guy?
SI-SCP	since your brother 's death , did you see him?
Source	a classic kind of friend is what she wants me to be.
Reference	she wants me to be like a classic kind of friend.
SGCP	my friend 's gon na be a classic friend.
SCPN	a classic type of boyfriend is what he wants to do.
SynTrans	it 's a classic friend of mine that she wants me.
GuiG	it 's me to be a classic friend of mine.
SI-SCP	he wants me to be a classic friend.

Table 6: Example paraphrases generated by each model on ParaNMT-Small Dataset.

Template	Paraphrase
Source	the balance between budget revenues and expenditure must be maintained.
(S (NP (PRP)) (VP (VBZ) (ADJP)) (.))	it is necessary to maintain the balance between budget revenue and expenditure.
(S (NP (NP) (PP)) (VP (MD) (VP)) (.))	the balance of budget revenue and expenditure must be kept.
(S (NP (NP) (PP)) (VP (VBZ) (ADJP)) (.))	the balance between budget revenue and expenditure is necessary.
(S (SBAR (WHADVP) (S)) (,) (NP (PRP)) (VP (VBZ) (ADJP)) (.))	when the budget revenue is recovered, it is necessary to maintain expenditure.

Table 7: Syntactic paraphrases generated by SI-SCP with retrieved templates. We show several successful and one failed (in blue) generations.