GCPG: A General Framework for Controllable Paraphrase Generation

Anonymous ACL submission

Abstract

Controllable paraphrase generation (CPG) incorporates various external conditions to obtain desirable paraphrases. However, existing works only highlight a special condition under two indispensable aspects of CPG (i.e., lexically and syntactically CPG) individually, lacking a unified circumstance to explore and analyze their effectiveness. In this paper, we propose a general controllable paraphrase generation framework (**GCPG**), which represents both lexical and syntactical conditions as text sequences and uniformly processes them in an encoder-decoder paradigm. Under GCPG, we reconstruct commonly adopted lexical condition (i.e., *Keywords*) and syntactical conditions (i.e., *Part-Of-Speech sequence*, *Constituent Tree*, *Masked Template* and *Sentential Exemplar*) and study the combination of the two types. In particular, for *Sentential Exemplar* condition, we propose a novel exemplar construction method — Syntax-Similarity based Exemplar (**SSE**). SSE retrieves a syntactically similar but lexically different sentence as the exemplar for each target sentence, avoiding exemplar-side words copying problem. Extensive experiments demonstrate that GCPG with SSE achieves state-of-the-art performance on two popular benchmarks. In addition, the combination of lexical and syntactical conditions shows the significant controllable ability of paraphrase generation, and these empirical results could provide novel insight to user-oriented paraphrasing.

1 Introduction

Paraphrase generation (Madnani and Dorr, 2010) refers to restating a given sentence into an alternative surface form while keeping the semantics unchanged. It is of long-standing interest (McKeown, 1983), with various applications such as question answering (Gan and Ng, 2019), machine translation (Mallinson et al., 2017), and sentence simplification (Martin et al., 2020). However, a sentence can be re-expressed in various surface forms. Lacking control might result in undesirable results (Gu et al., 2019).

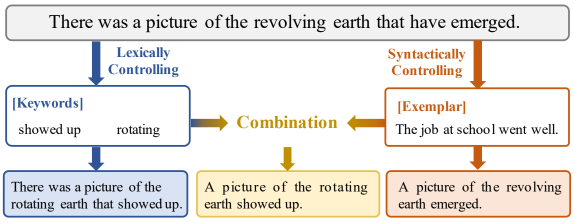

Figure 1: A toy example to explain what effect lexically controlling and syntactically controlling have on paraphrasing.

To obtain desirable surface forms, most recent works focus on controllable paraphrase generation (CPG) by incorporating external conditions. Existing efforts to CPG can be roughly divided into two types: lexically and syntactically CPG. Lexically CPG is concerned with *what to say*, which generates paraphrases that contain pre-specified keywords. As shown in Figure 1, a lexically CPG model needs to generate a paraphrase that contains the given keyword "showed up". To achieve it, a sequence-to-sequence model equipped with the copy mechanism is commonly used (Zeng et al., 2019). Different from lexically CPG, syntactically CPG concentrates on *how to say it*, generating a paraphrase that conforms to the syntax of a given exemplar (i.e., a sentence illustrating certain syntax patterns). Substantial efforts have been made on constructing syntactical features of the given exemplar. For example, Kumar et al. (2020) incorporate a full syntactic tree of the exemplar to guide paraphrasing; Bui et al. (2021) construct a masked template to direct generation by masking words with certain Part-of-Speech (POS) type of exemplar; Chen et al. (2019) directly use the sentential exemplar. Since sentential exemplars are only available for testing, they have to manufacture exemplars for training by replacing certain words from the target sentence. Despite the progress on the two types of conditions individually, *what to say* and *how to say it* are both aspects of vital importance for CPG (Kumar et al., 2020). Furthermore, there lacks a unified framework to study the effectiveness of these conditions and their joint utilization.

To fill this gap, we propose a **G**eneral **C**ontrollable **P**araphrase **G**eneration framework (GCPG) to jointly include both lexically and syntactically CPG in a unified model. The key idea is to reconstruct both lexical and syntactical conditions as text sequences and process them in a text-to-text encoder-decoder paradigm. This also allows GCPG to easily utilize the strong language modeling capacity of pre-trained language models (PLMs), which have demonstrated great potential (Bui et al., 2021) yet rarely been explored under the topic of CPG. For the lexical condition, we concatenate the pre-specified keywords as a sequence while exploring different methods to pre-specify keywords from rule-based to model-based. As for syntactical conditions, we reconstruct commonly used syntactic features as sequences, such as Linearised Constituent Tree (Iyyer et al., 2018) and masked template based on word mask (Bui et al., 2021). Besides the manufactured syntax features, we hypothesize that directly using the exemplar is more effective as it can benefit from the powerful sentence modeling capability of PLMs. To construct the exemplar for training, we propose a novel exemplar construction method as **S**yntax-**S**imilarity based **E**xemplar (SSE). Specifically, we use a sentence that is syntactically similar but lexically different from the target sentence, which is retrieved in a self-constructed exemplar dictionary based on the training set. This is different from existing methods that construct exemplar through modifying target sentences (Chen et al., 2019), alleviating exemplar-side words copying problem (Bui et al., 2021) brought by Chen et al. (2019).

We examine GCPG on two popular benchmark datasets. Those discussions include not only performances of different conditions and their combinations, but also the effectiveness of GCPG instantiated by different PLMs. Experiments demonstrate that GCPG consistently shows significant performances when tested by three different methods to pre-specify keywords. For syntactical CPG, GCPG with SSE obtains 13.95/24.31/18.64 ROUGE-1/2/L and 16.38 BLEU-4 over the previous state-of-the-art (SOTA) model (Bui et al., 2021). Besides, the combination of lexical and syntactical conditions show encouraging controllability of paraphrase generation in both quantitative and qualitative analysis. The main contributions are as follows:

- We propose **GCPG**, a general framework to jointly include both lexically and syntactically controllable paraphrasing. It is simple but effective, enabling flexible combinations of conditions by reconstructing them into text sequences and processing them in a text-to-text encoder-decoder paradigm. Those properties allow GCPG to easily adapt to mainstream pre-trained language models and utilize powerful language modeling capacity, which is rarely explored in CPG.
- We provide a novel exemplar construction method **SSE** under the syntactical condition. It allows GCPG to directly model syntax information from natural sentences without any manufactured syntax features, while alleviating the exemplar-side words copying problem.

2 Related Work

In this section, we summarize existing works on syntactically and lexically CPG. Syntactically CPG generates a paraphrase constrained by a pre-specified sentence of a certain syntax structure namely exemplar. However, the exemplar is only available during inference, resulting in a key challenge: obtaining manual exemplars for existing paraphrasing training datasets is prohibitively expensive. To address this, some of the previous works construct syntactical features from target sentences during training, such as *POS Tagging*, *Constituent Tree*, *mask template* as illustrated in Table 1. For instance, SCPN (Iyyer et al., 2018) makes the first attempt to introduce Linearised Constituent Tree (LCT) of target sentence into paraphrasing, where LCT is predicted based on predefined parse templates. Similarly, GuiG (Li et al., 2020) proposes two models to expand a partial template LCT and generate paraphrasing, respectively. Different from using LCT, SGCP (Kumar et al., 2020) introduces a graph encoder to encode the Constituent Tree of exemplar as the condition. Besides, *masked template* replaces several words of the exemplar with a special token to form a template as the condition. For example, BCPG (Liu et al., 2020b) follows BERT (Devlin et al., 2019) to randomly mask exemplar words, ParafraGPT (Bui

	Syntactical Condition			
Work	**POS Tagging**	**Constituent Tree**	**Masked Template**	**Sentential Exemplar**
SCPN (2018)	✓ (In Tree)	✓ (LCT Templates)	✗	✗
CGEN (2019)	✓ (In Exemplar)	✗	✗	✓ (Replace Words)
BCPG (2020b)	✗	✗	✓ (Randomly)	✗
GuiG (2020)	✗	✓ (Expanded LCT)	✗	✗
SGCP (2020)	✓ (In Tree)	✓ (Tree Structure)	✗	✗
ParafraGPT (2021)	✓ (In Word MT)	✗	✓ (Certain POS)	✗
GCPG	✓ (POS Sequence)	✓ (LCT)	✓ (Certain POS)	✓ (SSE)

Table 1: A comparison of different conditions under syntactically CPG. LCT: Linearised Constituent Tree. The proposed framework GCPG reconstructs them as text sequences and we have experimented with all four forms.

et al., 2021) further masks exemplar words with certain POS types. However, Chen et al. (2019) advocate to directly utilize the sentential exemplar (i.e., the sentence) as the condition, because they believe "any syntactically valid sentence is a valid exemplar". Since exemplar is only available in the testing set, they construct exemplar by replacing words of the target sentence with others that have the same POS type. Besides, lexically CPG constraints paraphrasing with pre-specified keywords, which is rarely explored but undoubtedly indispensable in CPG. Zeng et al. (2019) make the first attempt to integrate keywords with copy mechanism. Despite their progress, existing works only focus on a special condition under either lexically or syntactically CPG. In comparison, GCPG jointly includes lexically and syntactically CPG, flexibly combining conditions in a unified circumstance.

3 Methodology

3.1 GCPG Framework

Before introducing GCPG, we first give the definition of controllable paraphrase generation with external conditions. Given a source sentence $\boldsymbol{x}$ and a variety of conditions $\boldsymbol{c}$, the model generates paraphrase $\boldsymbol{y} = (y_1, y_2, ..., y_T)$ by:

$$p(\boldsymbol{y}|\boldsymbol{x}, \boldsymbol{c}) = \prod_{t=1}^{T} p(y_t|y_{<t}, \boldsymbol{x}, \boldsymbol{c}; \theta), \qquad (1)$$

where θ are the model parameters trained by maximizing the conditional likelihood of outputs in a parallel corpus. Given this definition, the forms of conditions $\boldsymbol{c}$ might be varied, such as pre-defined keywords and Constituent Parse Tree. To uniformly encode these conditions and investigate their effectiveness, we propose a general framework GCPG. GCPG contains a standard encoder-decoder paradigm, which allows any mainstream PLMs to adapt to this task rapidly. Meanwhile, GCPG can flexibly use the combinations of included conditions by concatenating them as one sequence with "[SEP]". As shown in Figure 2, the source sentence "No one's home ?" is concatenated with optional sequential conditions by the separator signal "[SEP]", then fed into the model. Afterward, the model auto-regressively generates "Is anyone home?" as the final result.

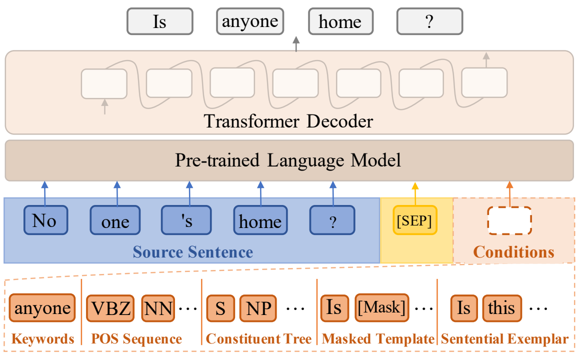

Figure 2: An overview of GCPG, the source sentence and separated condition are concatenated as input.

3.2 Conditions under GCPG

3.2.1 Syntactical Condition

Syntactically CPG requests a syntax exemplar to constrain the syntax structure of paraphrase. However, exemplars are only available in the testing set of existing paraphrasing datasets. To train a syntactically CPG model, we construct a syntactical condition based on the target sentences in the training set. During inference, we apply the same strategy to obtain the corresponding syntactical conditions from exemplars in the testing set. We explore four syntactical conditions in this work, as follows:

POS Tagging is one of most simple solutions in

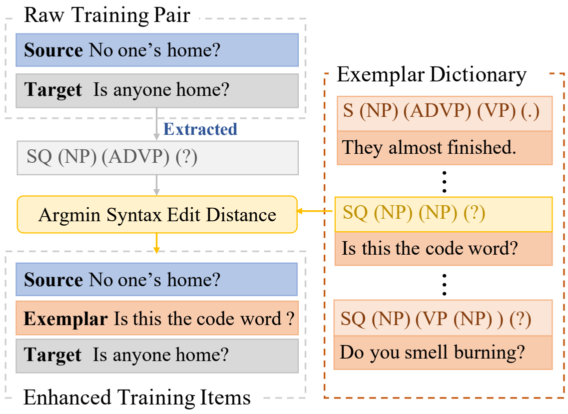

Figure 3: An overview of SSE. We take Truncated LCT as the sequential syntax structure here.

modeling the syntax structure (Cutting et al., 1992), which could be effectively implemented and show promising performance in various NLP tasks (Yang et al., 2021). We investigate POS Tagging as an independent condition, which is rarely explored in CPG. In detail, we extract POS sequence of target sentence by CoreNLP[1] as the condition. To learn these POS signals with PLMs, we regard these POS tokens as special ones and add them into the word vocabulary of PLMs.

Constituent Tree is a widely used condition for syntax controlling while paraphrasing. Here, we explore two kinds of LCT, i.e., full-fledged LCT and Truncated LCT. For the full-fledged LCT condition, we extract the complete sequential Constituent Tree from the target sentence for training and exemplar for testing, based on the off-the-shelf tools of CoreNLP. We further explore the Truncated LCT condition, which is the sequence that removing POS-level tokens in full-fledged LCT. Compared with full-fledged LCT, Truncated LCT drastically shortens the input length.

Masked Template is first introduced in Liu et al. (2020b), which randomly masks words of the target sentence to form a syntax template as the condition. To verify the effectiveness of this method in GCPG circumstance, we follow the current SOTA Bui et al. (2021) to construct a masked template by substituting all nouns, adjectives, adverbs, and verbs with a special token in the exemplar. Similarly, this strategy is applied to the target sentences during training and the given exemplars during inference.

Sentential Exemplar is the most straightforward way for syntactically CPG, which directly uses the sentential exemplar as the condition. In contrast to the above three syntactical conditions, Sentential Exemplar uses natural sentences to represent desirable syntax structure, without introducing any special token which does not appear during PLMs pre-training. We argue that this way can make better use of PLMs. However, the previous method (Chen et al., 2019) suffers from the exemplar-side words copying problem during testing, which might be caused by the noticeable words overlap with the target sentence in constructing sentential exemplar during training. To alleviate this problem, we propose **S**yntax-**S**imilarity based **E**xemplar (**SSE**) to enhance sentential exemplar condition.

An overview of our SSE method is demonstrated in Figure 3. To alleviate the exemplar-side words copying issue, the proposed SSE constructs Sentential Exemplar by retrieving a syntactically similar but lexically different sentence for each target sentence during training. To achieve that, we construct an exemplar dictionary that contains the syntactical key-value mapping from the syntax structure $\boldsymbol{k}$ to its corresponding natural sentence $\boldsymbol{v}$. Each syntactical key $\boldsymbol{k} \in \boldsymbol{K}$ is a Truncated LCT sequence, and its value is a randomly selected natural sentence that can be assigned to this Truncated LCT sequence. During training, given a data pair $\langle \boldsymbol{x}, \boldsymbol{y} \rangle$ and the Truncated LCT $\boldsymbol{s}$ of $\boldsymbol{y}$, we select a syntactical key $\boldsymbol{k}^*$ by calculating the syntax edit distance D_{syn} between $\boldsymbol{s}$ and each syntactical key in the exemplar dictionary, which can be formulated as:

$$\begin{aligned} \boldsymbol{k}^* &= \arg\min(D_{\text{syn}}(\boldsymbol{s}, \boldsymbol{k})) \\ &= \arg\min_{\boldsymbol{k} \in \boldsymbol{K}}\left(\frac{\text{LevEdit}(\boldsymbol{s}, \boldsymbol{k})}{\max(|\boldsymbol{s}|, |\boldsymbol{k}|)}\right), \end{aligned} \tag{2}$$

where $\text{LevEdit}(\cdot)$ denotes the token-level Levenshtein edit distance between two sequences and $|\cdot|$ denotes the token-level length of the sequence. We assign the corresponding sentence $\boldsymbol{v}^*$, which is related to $\boldsymbol{k}^*$, as the training exemplar.

Lexical Condition Lexically CPG uses pre-specified keywords to constrain paraphrasing, which requires a paraphrasing dataset containing $\langle$*sentence*, *keywords*, *paraphrase*$\rangle$ triples. Because the original dataset is formatted as $\langle$*sentence*, *paraphrase*$\rangle$, we need to pre-specify keywords for each data item. Following Zeng et al. (2019), we automatically extract keywords from the target sentence as the condition in the training stage. Besides, as also lacking manual keywords for each

[1] https://stanfordnlp.github.io/CoreNLP/index.html

testing pair, we carry out two strategies for inference. On the one hand, we directly extract keywords from references as conditions following Zeng et al. (2019). On another, a standard sequence-to-sequence model is used to predict target keywords only from source sentences as conditions while testing, as described in Liu et al.(2020a). Specifically, we investigate three representative keyword extraction methods to verify the effectiveness of GCPG, including rule-based TF-IDF, TextRank (Mihalcea and Tarau, 2004), and model-based KeyBERT (Grootendorst, 2020). Each method filters out the stop words and punctuation, and guarantees the extracted keywords do not appear in the corresponding source sentence. The maximum number of keywords is set to 3. Besides, we use a special token "[NONE]" when there are no keywords extracted.

4 Experiments

In this section, we individually evaluate syntactically and lexically conditions under GCPG, then examine their combinations. Finally, detailed analyses on properties of GCPG are provided.

Datasets Following previous works (Kumar et al., 2020; Bui et al., 2021), we evaluate GCPG on two datasets: (1) **ParaNMT-small** (Chen et al., 2019) is a subset of ParaNMT-50M dataset (Wieting and Gimpel, 2018), which is collected via back-translation referring to English sentences. It contains 500K training pairs formatted as ⟨sentence, paraphrase⟩, and 1.3K manually labeled data triples formatted as ⟨*sentence*, *exemplar*, *paraphrase*⟩ (0.8K for testing and 0.5K for validation). In each triple, *exemplar* is a sentence that has the same syntax as *paraphrase* but is semantically different from *sentence*. (2) **QQP-Pos** (Kumar et al., 2020) is selected from Quora Question Pairs (QQP) dataset. It contains about 140K training pairs and 3K/3K data triples for testing/validation. The format of dataset is the same as ParaNMT-small.

4.1 Syntactically Controllable Paraphrasing

We explore four syntactical conditions reconstructed by GCPG on the ParaNMT-small dataset, then compare SSE with baselines on two datasets.

Baselines We first choose two direct return-input baselines as dataset quality indicators: (1) *Source-as-Output* copies inputs as outputs. (2) *Exemplar-as-Output* regards exemplars as outputs. Next, we evaluate the following text generation models, while exploring performances of respectively instantiating GCPG with them in § 4.3. (3) *Transformer* (Vaswani et al., 2017), the conventional version in the original paper. (4) *BART* (Lewis et al., 2020) has a denoising autoencoder for pre-training sequence-to-sequence models, and BART-large[2] is used. (5) *ProphetNet* (Qi et al., 2020) is a pre-training model with a self-supervised objective, and ProphetNet-large is used. Finally, we compare GCPG with mainstream competitive models as follows. (6) *SCPN* (Iyyer et al., 2018) has two encoders to encode source sentence and LCT separately, then constrain generation with soft attention mechanism[3]. (7) *CGEN* (Chen et al., 2019) encodes exemplars into latent vector to guide paraphrasing[4]. (8) *SGCP* (Kumar et al., 2020) uses a graph encoder to process the exemplar Constituent Trees as the condition[5]. (9) *ParafraGPT* (Bui et al., 2021) masks words with certain POS types in the target sentence as condition, then builds a paraphrasing generator based on a pre-trained GPT2.

Syntactical Conditions We first examine conditions with manufactured syntax features, including (10) *POS Sequence*, (11) *LCT-Truncated* is the *LCT* sequence without POS-level information, (12) *LCT* is the full-fledged Linearised Constituent Tree sequence, and (13) *Masked Template*. Then, two implementations of SSE are evaluated: (14) *SSE-POS Sequence* uses *POS Sequence* to measure syntax similarity, and (15) *SSE-LCT-Truncated* uses *LCT-Truncated* as measurement.

Implementation and Hyper-parameters All GCPG models are instantiated by ProphetNet-large (Qi et al., 2020), which are implemented with Fairseq[6]. We employ the original hyper-parameter setting of ProphetNet-large[7] to train GCPG. During inference, the beam size and length penalty are set to 4 and 1.2 following Bui et al. (2021).

Metrics Following previous works (Iyyer et al., 2018; Bui et al., 2021), we evaluate generating results on six metrics, including BLEU-4 (Papineni et al., 2002), ROUGE-1 (R-1), ROUGE-2 (R-2), ROUGE-L (R-L) (Lin, 2004), Meteor (MTR) (Denkowski and Lavie, 2014), and

[2] https://github.com/pytorch/fairseq/tree/master/examples/bart
[3] https://github.com/miyyer/scpn
[4] https://github.com/mingdachen/syntactic-template-generation
[5] https://github.com/malllabiisc/SGCP
[6] https://github.com/pytorch/fairseq
[7] https://github.com/microsoft/ProphetNet

Model	iBLEU ↑	B-R ↑	R-1 / R-2 / R-L ↑	MTR ↑	BS ↑	TED ↓
			ParaNMT-small			
(1) Source-as-Output	-17.05	18.50	23.10 / 47.70 / 12.00	28.80	86.20	12.00
(2) Exemplar-as-Output	2.31	3.30	24.40 / 7.50 / 29.10	12.10	74.20	5.90
(3) Transformer	4.72	14.66	51.05 / 26.88 / 51.32	30.67	91.30	12.71
(4) BART	6.08	17.78	52.37 / 27.02 / 51.52	31.57	91.99	11.92
(5) ProphetNet	4.67	18.46	55.29 / 31.17 / 55.18	32.42	92.32	11.78
(6) SCPN (2018)	–	6.40	30.30 / 11.20 / 34.60	14.60	73.70	9.10
(7) CGEN (2019)	8.14	13.60	44.80 / 21.00 / 48.30	24.80	79.50	**6.70**
(8) SGCP (2020)	6.95	16.40	49.60 / 22.90 / 50.50	27.20	80.50	6.80
(9) ParafraGPT (2021)	8.61	14.54	49.67 / 22.42 / 51.29	27.83	90.78	8.22
(10) GCPG (POS Sequence)	11.96	19.97	56.20 / 32.36 / 58.99	32.68	92.57	8.45
(11) GCPG (LCT-Truncated)	12.74	22.54	59.98 / 36.81 / 62.61	37.04	93.39	8.34
(12) GCPG (LCT)	11.92	19.52	55.75 / 30.54 / 58.88	31.35	92.42	7.84
(13) GCPG (Masked Template)	9.52	16.85	53.60 / 27.96 / 56.31	31.84	92.21	8.84
(14) GCPG (SSE-POS Sequence)	10.07	23.82	60.93 / 37.36 / 61.98	36.15	91.55	8.94
(15) GCPG (SSE-LCT-Truncated)	**12.32**	**26.24**	**63.62 / 40.76 / 64.98**	**39.79**	**93.86**	8.27
			QQP-Pos			
(16) Source-as-Output	-17.96	17.20	51.90 / 26.20 / 52.90	31.10	84.90	16.20
(17) Exemplar-as-Output	10.64	16.80	38.20 / 20.50 / 43.20	17.60	78.20	4.80
(18) Transformer	7.63	23.44	54.58 / 30.48 / 56.63	32.60	93.18	11.84
(19) BART	3.14	23.07	56.43 / 32.12 / 57.64	34.26	93.58	13.05
(20) ProphetNet	6.43	25.79	58.40 / 34.52 / 59.98	35.75	93.88	11.74
(21) SCPN (2018)	–	15.60	40.60 / 20.50 / 44.60	19.60	77.60	9.10
(22) CGEN (2019)	17.60	29.94	58.53 / 37.42 / 61.74	32.90	92.82	6.43
(23) SGCP (2020)	19.97	38.00	68.10 / 45.70 / 70.20	41.30	94.53	6.80
(24) ParafraGPT (2021)	21.19	35.86	66.71 / 43.70 / 68.94	40.26	94.54	6.11
(25) GCPG (SSE-LCT-Truncated)	**28.10**	**50.62**	**77.32 / 59.04 / 79.02**	**51.45**	**96.49**	**5.02**

Table 2: Results of different syntactical conditions and comparisons with baselines on ParaNMT-small and QQP-Pos datasets. B-R: BLEU-R. R-1:ROUGE-1. R-2: ROUGE-2. R-L: ROUGE-L. MTR:METEOR. BS:BERTScore. ↑ means higher score is better where ↓ is exactly the opposite. The highest numbers are in **bold**.

BERTScore (BS) (Zhang et al., 2020). Besides, *Source-as-Output* will also get a high BLEU score and BERTScore, we introduce iBLEU (Sun and Zhou, 2012) for more precise evaluation. As a variant of BLEU, iBLEU considers both fidelity to *reference* and diversification from *input*:

$$\begin{aligned} \text{iBLEU} &= \alpha\text{BLEU-R} - (1 - \alpha\text{BLEU-S}), \\ \text{BLEU-R} &= \text{BLEU-4}\,(\textit{output}, \textit{reference}), \\ \text{BLEU-S} &= \text{BLEU-4}\,(\textit{output}, \textit{input}), \end{aligned} \tag{3}$$

where the constant α is set to 0.7, as in the original paper. Finally, for syntactical condition evaluation, we follow Kumar et al. (2020) to calculate Tree-Edit Distance (TED)[8] between the Constituency Parse Trees of both *output* and *reference*.

Results As shown in Table 2, the main conclusions are: (1) SSE consistently and significantly outperforms conditions that constructed with manufactured syntax features (Rows 14-15 vs. Rows 10-13). (2) GCPG with SSE gets significant improvement over the previous SOTA (Row 15/25 vs. Row 14/24). (3) All syntactical conditions reconstructed in GCPG outperform baselines (Rows 10-15 vs. Rows 6-9), demonstrating the superiority of GCPG paradigm.

[8]We use the evaluation tool implemented by SGCP.

4.2 Lexically Controllable Paraphrasing

As mentioned in § 3.2, we use three different keyword extraction methods to pre-specify keywords and comprehensively evaluate the GCPG: (1) *TF-IDF* (2) *TextRank* (Mihalcea and Tarau, 2004), and (3) *KeyBERT* (Grootendorst, 2020). Meanwhile, we follow the implementation settings in § 4.1.

Metrics For lexical condition, it should be noted that there is a lack of the explicit request of desirable keywords in the testing set. A generated paraphrase hinted by model predicted keywords might get a low score in BLEU, although humans consider it reasonable. This is because paraphrasing models might focus on keywords that are not

Condition	iBLEU ↑	B-R ↑	R-1 / R-2 / R-L ↑	MTR ↑	BS ↑	TED ↓
Keywords Extraction, GCPG instantiated by ProphetNet						
(1) GCPG (None)	4.67	18.46	55.29 / 31.17 / 55.18	32.42	92.32	11.78
(2) GCPG (TF-IDF)	10.07	23.04	61.92 / 38.68 / 61.71	36.97	92.86	10.79
(3) GCPG (TextRank)	8.16	19.63	56.04 / 32.08 / 56.54	33.60	92.45	12.47
(4) GCPG (KeyBERT)	11.03	24.12	60.92 / 38.00 / 61.14	35.41	92.79	10.26
(5) GCPG (KeyBERT (Upper Bound))	16.06	28.64	67.81 / 43.99 / 66.30	40.27	93.44	9.98
Keywords (KeyBERT) + Syntactical Condition, GCPG instantiated by ProphetNet						
(6) GCPG (KeyBERT + POS Sequence)	15.10	25.22	62.96 / 39.04 / 65.32	36.42	90.96	8.01
(7) GCPG (KeyBERT + LCT-Truncated)	15.38	26.80	66.07 / 43.52 / 68.07	39.53	90.56	8.08
(8) GCPG (KeyBERT + LCT)	14.47	23.52	61.92 / 36.33 / 64.38	34.73	92.74	8.00
(9) GCPG (KeyBERT + Mask Template)	12.13	20.98	58.83 / 33.58 / 61.01	35.02	92.67	8.44
(10) GCPG (KeyBERT + SSE-POS)	15.67	**31.02**	66.85 / 45.30 / 68.48	40.12	90.39	7.95
(11) GCPG (KeyBERT + SSE-LCT-Truncated)	**15.73**	30.92	**68.40 / 46.73 / 69.93**	**41.98**	**94.34**	7.95
Condition (11), GCPG instantiated by Different Models						
(12) GCPG-LS (Transformer)	11.22	21.26	60.94 / 37.10 / 62.52	35.77	92.67	9.21
(13) GCPG-LS (BART)	14.23	26.80	66.32 / 44.97 / 67.86	40.60	93.90	9.51
(14) GCPG-LS (ProphetNet)	15.73	30.92	68.40 / 46.73 / 69.93	41.98	94.34	7.95

Table 3: Performance of different conditions and combinations under GCPG on ParaNMT-small.

consistent with the single reference. Therefore, we evaluate GCPG in three settings. First, following Liu et al.(2020a), we use a keywords prediction model to generate top-k groups of keywords, which are fed into GCPG to generate k paraphrases. Then the sentence that has the highest BLEU with the reference is selected as the final output. k is set to 4 as well as beam size. Note that we use this setting to report the final results unless otherwise specified. Second, we further conduct human evaluations on the keyword condition based on *KeyBERT* (The details are in § 4.3). We denote it as "GCPG-L (k=1)". Here "k=1" means GCPG only produces one paraphrase for each input, constrained by the top-1 set of keywords produced by *KeyBERT*. Third, following Zeng et al. (2019), we directly extract keywords from references as the condition, marked with "(Upper Bound)".

Results As shown in the first five rows of Table 3, *KeyBERT* outperforms other two keyword extraction methods. Besides, GCPG with keyword condition significantly performs better than GCPG without keyword condition, which verifies the lexically controllable ability of our GCPG.

4.3 Combinations

We first discuss combinations of lexical and syntactical conditions, and then evaluate GCPG instantiated by different PLMs. To facilitate the description, we define that "GCPG-L" denotes GCPG with the keyword condition extracted by *KeyBERT*, "GCPG-S" is GCPG with the *SSE-LCT-Truncated* condition, and "GCPG-LS" indicates the combination of conditions in "GCPG-L" and "GCPG-S". Meanwhile, GCPG is also instantiated by ProphetNet-large.

Metrics We follow the metrics in § 4.1, yet the automatic evaluations can not fully capture the fluency and the quality of the generation results on CPG. Especially for TED, as the ParaNMT-small contains various noise data points, it is optimistic to assume that the corresponding constituency parse tree could be well aligned (Kumar et al., 2020). Therefore, we conduct human evaluation on both two datasets following Kumar et al.(2020). 100 test samples are randomly selected from each dataset. Then, 5 crowdsource evaluators are shown a source sentence and the corresponding reference, then asked to rate model results in three categories: whether the paraphrase remains **loyalty** to the source sentence, the **fluency** of paraphrase, and **syntax** similarity with gold reference. Scores are ranged from 1 to 4, and the higher score is better.

Results As shown in Table 3, the main conclusions are: (1) Combinations of lexical and syntactical conditions get consistently further improvements compared with employing lexical condition individually (Rows 6-11 vs. Row 4). (2) GCPG can utilize the strong language modeling capacity of mainstream PLMs and show encouraging performances (Row 12-13 vs. Row 14). Then, we illustrate human evaluations in Table 4. GCPG with lexical condition (GCPG-L (k=1)) outperforms baselines in meaning and fluency, yet poor in syntax similar-

Model	Loyalty	Fluency	Syntax	All
ParaNMT-small				
CGEN	1.47	2.13	1.81	5.41
ParafraGPT	1.86	2.42	2.05	6.33
GCPG-L (k=1)	2.94	3.63	2.29	8.86
GCPG-LS (k=1)	3.09	3.51	2.46	**9.06**
QQP-Pos				
CGEN	1.72	2.52	2.22	6.46
ParafraGPT	2.43	2.91	2.61	7.95
GCPG-L (k=1)	3.00	3.54	2.43	8.97
GCPG-LS (k=1)	2.97	3.43	2.81	**9.21**

Table 4: Results of Human evaluation.

Model	BLEU-Exemplar ↓	
	ParaNMT-small	QQP-Pos
ParafraGPT	7.32	24.31
GCPG-S	2.63	23.17
Reference	3.30	16.80

Table 5: GCPG can significantly reduce BLEU-Exemplar score compared with previous SOTA.

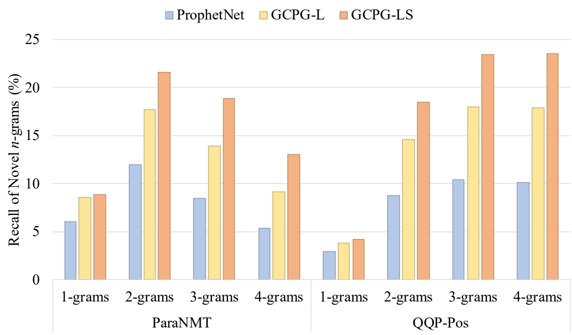

Figure 4: Recall of novel n-grams results.

Input	A powerful restorative energy emerges out of love.
Exemplar	There's one thing that makes me feel normal.
Reference	There is a powerful healing energy that emanates from loving.
GCPG-L	A powerful healing energy comes out of love. [healing]
GCPG-LS	There's a powerful healing energy that comes out of love. [healing]
Input	We 'd climb the mountain and make a house there ?
Exemplar	Will we have a list of six demands ?
Reference	Will we build a house in the mountain?
GCPG-L	Would we climb a mountain and build a house? [build]
GCPG-LS	Will we build a house in the mountain ? [build]

Figure 5: Samples of paraphrases. Words in “[]” are offered by our keywords prediction model based on KeyBERT. We highlight different parts for better view.

ity. More importantly, the combination of lexical and syntactical conditions (GCPG-LS (k=1)) shows significantly improvements on all three scores.

4.4 Analyses and Discussions

We conduct discussions to shed light on other interesting properties of GCPG. For the lack of space, we take discussions with GCPG instantiated by ProphetNet-large.

Exemplar-side Words Copying Problem We calculate BLEU-4 between model outputs and exemplars. As shown in Table 5, GCPG with SSE (i.e., GCPG-S) can significantly reduce BLEU-Exemplar comparing with ParafraGPT, gets 4.69 / 1.14 improvements on two datasets, demonstrating that SSE effectively alleviates this problem.

Generating Novel Grams Following Dou et al.(2021), we further investigate generating novel expressions under CPG settings, which is also important for paraphrasing. To address this issue, the number of novel n-grams is counted in the model output. Specifically, these n-grams appear in gold references but not in source sentences. After normalized by the total number of n-grams, we calculate the recall of novel n-grams. It can be seen that GCPG indeed generates novel expressions from Figure 4. The combination version GCPG-LS gets the best result, which means combination of two types of conditions may improve the lexical diversification from the input.

Case Studies The qualitative effect of the lexical and syntactical conditions on the model output is also of interest. To intuitively display the effects of conditions, we show some paraphrasing results in Figure 5. In detail, GCPG-L can generate sentence “A powerful healing energy comes out of love.” that contain pre-specified keywords “[healing]". However, lexical condition provides less information about syntactical controlling. In comparison, GCPG-LS shows better performances on both controllability of lexical items and syntax.

5 Conclusions

In this paper, we propose a general framework GCPG, enabling flexibly combine lexical and syntactical conditions and exploring their mutual effectiveness. Under GCPG, we provide SSE that allows GCPG to directly model syntax information from natural sentences and better utilize PLMs. As we tentatively give a successful implementation of leveraging two types of conditions in a unified circumstance, such paradigm deserves a closer and more detailed exploration. In the future, we will investigate to uniformly represent these conditions in a more superior way.

References

Tien-Cuong Bui, Van-Duc Le, Hai-Thien To, and Sang-Kyun Cha. 2021. Generative pre-training for paraphrase generation by representing and predicting spans in exemplars. In *IEEE BigComp*, pages 83–90. IEEE.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019. Controllable paraphrase generation with a syntactic exemplar. In *ACL*, pages 5972–5984. ACL.

Douglass Cutting, Julian Kupiec, Jan Pedersen, and Penelope Sibun. 1992. A practical part-of-speech tagger. In *Third Conference on Applied Natural Language Processing*, pages 133–140.

Michael J. Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *WMT-ACL*, pages 376–380. ACL.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: pre-training of deep bidirectional transformers for language understanding. In *NAACL*, pages 4171–4186. ACL.

Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. 2021. Gsum: A general framework for guided neural abstractive summarization. In *NAACL*, pages 4830–4842. ACL.

Wee Chung Gan and Hwee Tou Ng. 2019. Improving the robustness of question answering systems to question paraphrasing. In *ACL*, pages 6065–6075. ACL.

Maarten Grootendorst. 2020. Keybert: Minimal keyword extraction with bert.

Yunfan Gu, Yang Yuqiao, and Zhongyu Wei. 2019. Extract, transform and filling: A pipeline model for question paraphrasing based on template. In *W-NUT*, pages 109–114.

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

Ashutosh Kumar, Kabir Ahuja, Raghuram Vadapalli, and Partha P. Talukdar. 2020. Syntax-guided controlled generation of paraphrases. *Trans. Assoc. Comput. Linguistics*, 8:330–345.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *ACL*, pages 7871–7880. ACL.

Yinghao Li, Rui Feng, Isaac Rehg, and Chao Zhang. 2020. Transformer-based neural text generation with syntactic guidance. *CoRR*, abs/2010.01737.

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

Dayiheng Liu, Yeyun Gong, Yu Yan, Jie Fu, Bo Shao, Daxin Jiang, Jiancheng Lv, and Nan Duan. 2020a. Diverse, controllable, and keyphrase-aware: A corpus and method for news multi-headline generation. In *EMNLP*, pages 6241–6250. ACL.

Mingtong Liu, Erguang Yang, Deyi Xiong, Yujie Zhang, Chen Sheng, Changjian Hu, Jinan Xu, and Yufeng Chen. 2020b. Exploring bilingual parallel corpora for syntactically controllable paraphrase generation. In *IJCAI*, pages 3955–3961. ijcai.org.

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

Jonathan Mallinson, Rico Sennrich, and Mirella Lapata. 2017. Paraphrasing revisited with neural machine translation. In *EACL*, pages 881–893. ACL.

Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2020. Muss: Multilingual unsupervised sentence simplification by mining paraphrases. *arXiv preprint arXiv:2005.00352*.

Kathleen R. McKeown. 1983. Paraphrasing questions using given and new information. *Am. J. Comput. Linguistics*, 9(1):1–10.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into text. In *EMNLP*, pages 404–411.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. ACL.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. Prophetnet: Predicting future n-gram for sequence-to-sequence pre-training. In *EMNLP*, volume EMNLP 2020 of *Findings of ACL*, pages 2401–2410. ACL.

Hong Sun and Ming Zhou. 2012. Joint learning of a dual SMT system for paraphrase generation. In *ACL*, pages 38–42. ACL.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NeurIPS*, 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. Association for Computational Linguistics.

Kexin Yang, Wenqiang Lei, Dayiheng Liu, Weizhen Qi, and Jiancheng Lv. 2021. Pos-constrained parallel decoding for non-autoregressive generation. In *ACL*, pages 5990–6000. ACL.

Daojian Zeng, Haoran Zhang, Lingyun Xiang, Jin Wang, and Guoliang Ji. 2019. User-oriented paraphrase generation with keywords controlled network. *IEEE Access*, 7:80542–80551.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with BERT. In *ICLR*. OpenReview.net.