Neural Syntactic Preordering for Controlled Paraphrase Generation

Tanya Goyal and **Greg Durrett**
Department of Computer Science
The University of Texas at Austin
`tanyagoyal@utexas.edu, gdurrett@cs.utexas.edu`

Abstract

Paraphrasing natural language sentences is a multifaceted process: it might involve replacing individual words or short phrases, local rearrangement of content, or high-level restructuring like topicalization or passivization. Past approaches struggle to cover this space of paraphrase possibilities in an interpretable manner. Our work, inspired by pre-ordering literature in machine translation, uses syntactic transformations to softly "reorder" the source sentence and guide our neural paraphrasing model. First, given an input sentence, we derive a set of feasible syntactic rearrangements using an encoder-decoder model. This model operates over a partially lexical, partially syntactic view of the sentence and can reorder big chunks. Next, we use each proposed rearrangement to produce a sequence of position embeddings, which encourages our final encoder-decoder paraphrase model to attend to the source words in a particular order. Our evaluation, both automatic and human, shows that the proposed system retains the quality of the baseline approaches while giving a substantial increase in the diversity of the generated paraphrases.[1]

1 Introduction

Paraphrase generation (McKeown, 1983; Barzilay and Lee, 2003) has seen a recent surge of interest, both with large-scale dataset collection and curation (Lan et al., 2017; Wieting and Gimpel, 2018) and with modeling advances such as deep generative models (Gupta et al., 2018; Li et al., 2019). Paraphrasing models have proven to be especially useful if they expose control mechanisms that can be manipulated to produce diverse paraphrases (Iyyer et al., 2018; Chen et al., 2019b; Park et al., 2019), which allows these models to be employed for data augmentation (Yu et al., 2018) and

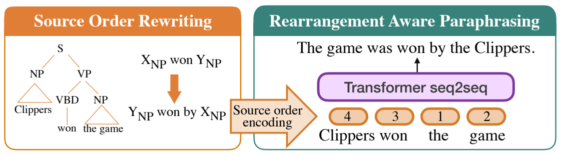

Figure 1: Overview of our paraphrase model. First, we choose various pairs of constituents to abstract away in the source sentence, then use a neural transducer to generate possible reorderings of the abstracted sentences. From these, we construct a guide reordering of the input sentence which then informs the generation of output paraphrases.

adversarial example generation (Iyyer et al., 2018). However, prior methods involving syntactic control mechanisms do not effectively cover the space of paraphrase possibilities. Using syntactic templates covering the top of the parse tree (Iyyer et al., 2018) is inflexible, and using fully-specified exemplar sentences (Chen et al., 2019b) poses the problem of how to effectively retrieve such sentences. For a particular input sentence, it is challenging to use these past approaches to enumerate the set of reorderings that make sense *for that sentence.*

In this paper, we propose a two-stage approach to address these limitations, outlined in Figure 1. First, we use an encoder-decoder model (SOW, for Source Order reWriting) to apply transduction operations over various abstracted versions of the input sentence. These transductions yield possible reorderings of the words and constituents, which can be combined to obtain multiple feasible rearrangements of the input sentence. Each rearrangement specifies an order that we should visit words of the source sentence; note that such orderings could encourage a model to passivize (visit the object before the subject), topicalize, or reorder clauses. These orderings are encoded for our encoder-decoder paraphrase model (REAP, for REarrangement Aware Paraphrasing) by way of po-

[1]Data and code are available at `https://github.com/tagoyal/sow-reap-paraphrasing`

sition embeddings, which are added to the source sentence encoding to specify the desired order of generation (see Figure 2). This overall workflow is inspired by the pre-ordering literature in machine translation (Xia and McCord, 2004; Collins et al., 2005); however, our setting explicitly requires entertaining a *diverse* set of possible orderings corresponding to different paraphrasing phenomena.

We train and evaluate our approach on the large-scale English paraphrase dataset PARANMT-50M (Wieting and Gimpel, 2018). Results show that our approach generates considerably more diverse paraphrases while retaining the quality exhibited by strong baseline models. We further demonstrate that the proposed syntax-based transduction procedure generates a feasible set of rearrangements for the input sentence. Finally, we show that position embeddings provide a simple yet effective way to encode reordering information, and that the generated paraphrases exhibit high compliance with the desired reordering input.

2 Method

Given an input sentence $\mathbf{x} = \{x_1, x_2, \ldots, x_n\}$, our goal is to generate a set of structurally distinct paraphrases $Y = \{\mathbf{y}^1, \mathbf{y}^2, \ldots, \mathbf{y}^k\}$. We achieve this by first producing k diverse reorderings for the input sentence, $R = \{\mathbf{r}^1, \mathbf{r}^2, \ldots, \mathbf{r}^k\}$, that guide the generation order of each corresponding $\mathbf{y}$. Each reordering is represented as a permutation of the source sentence indices.

Our method centers around a sequence-to-sequence model which can generate a paraphrase roughly respecting a particular ordering of the input tokens. Formally, this is a model $P(\mathbf{y} \mid \mathbf{x}, \mathbf{r})$. First, we assume access to the set of target reorderings R and describe this rearrangement aware paraphrasing model (REAP) in Section 2.2. Then, in Section 2.3, we outline our reordering approach, including the source order rewriting (SOW) model, which produces the set of reorderings appropriate for a given input sentence $\mathbf{x}$ during inference ($\mathbf{x} \to R$).

2.1 Base Model

The models discussed in this work build on a standard sequence-to-sequence transformer model (Vaswani et al., 2017) that uses stacked layers of self-attention to both encode the input tokens $\mathbf{x}$ and decode the corresponding target sequence $\mathbf{y}$. This model is pictured in the gray block of Figure 2. Throughout this work, we use byte pair encoding

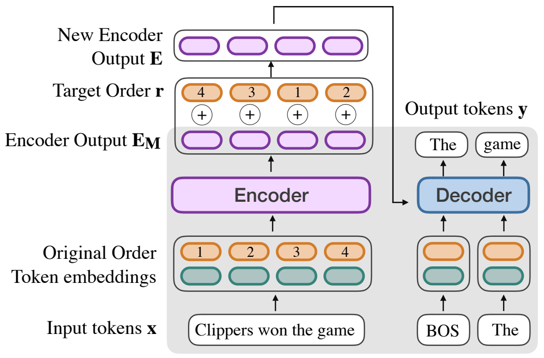

Figure 2: Rearrangement aware paraphrasing (REAP) model. The gray area corresponds to the standard transformer encoder-decoder system. Our model adds position embeddings corresponding to the target reordering to encoder outputs. The decoder attends over these augmented encodings during both training and inference.

(BPE) (Sennrich et al., 2016) to tokenize our input and output sentences. These models are trained in the standard way, maximizing the log likelihood of the target sequence using teacher forcing. Additionally, in order to ensure that the decoder does not attend to the same input tokens repeatedly at each step of the decoding process, we include a coverage loss term, as proposed in See et al. (2017).

Note that since the architecture of the transformer model is non-recurrent, it adds position embeddings to the input word embeddings in order to indicate the correct sequence of the words in both $\mathbf{x}$ and $\mathbf{y}$ (see Figure 2). In this work, we propose using an additional set of position embeddings to indicate the desired order of words during generation, described next.

2.2 Rearrangement aware Paraphrasing Model (REAP)

Let $\mathbf{r} = \{r_1, r_2, \ldots, r_n\}$ indicate the target reordering corresponding to the input tokens $\mathbf{x}$. We want the model to approximately attend to tokens in this specified order when generating the final output paraphrase. For instance, in the example in Figure 1, the reordering specifies that when producing the paraphrase, the model should generate content related to *the game* before content related to *Clippers* in the output. In this case, based on the rearrangement being applied, the model will most likely use passivization in its generation, although this is not strictly enforced.

The architecture for our model $P(\mathbf{y} \mid \mathbf{x}, \mathbf{r})$ is outlined in Figure 2. Consider an encoder-decoder architecture with a stack of M layers in the encoder

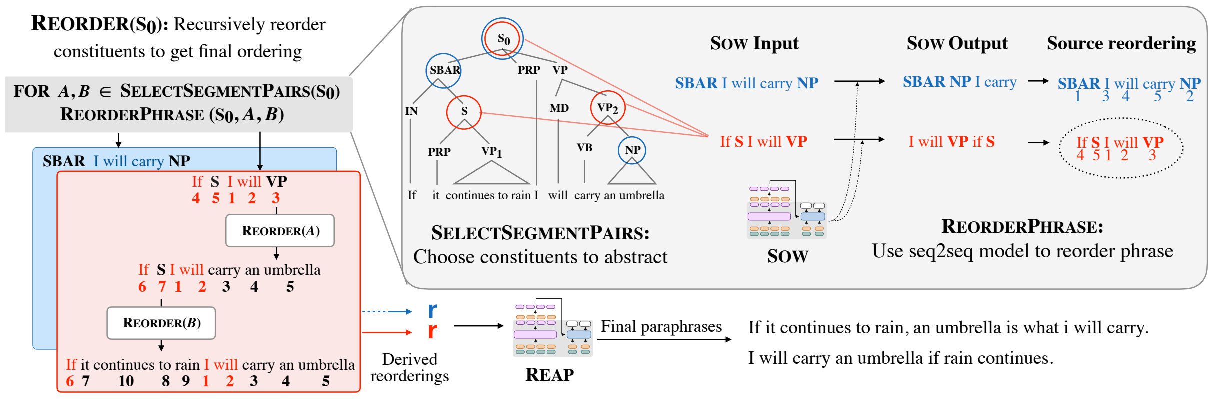
REORDER(S0): Recursively reorder constituents to get final ordering
FOR A, B ∈ SELECTSEGMENTPAIRS(S0)
REORDERPHRASE (S0, A, B)
SBAR I will carry NP
If S I will VP
4 5 1 2 3
REORDER(A)
If S I will carry an umbrella
6 7 1 2 3 4 5
REORDER(B)
If it continues to rain I will carry an umbrella
6 7 10 8 9 1 2 3 4 5
r
r
Derived reorderings
REAP
Final paraphrases
If it continues to rain, an umbrella is what i will carry.
I will carry an umbrella if rain continues.
S0
SBAR
PRP
VP
IN
S
MD
VP2
PRP
VP1
VB
NP
If it continues to rain I will carry an umbrella
SELECTSEGMENTPAIRS:
Choose constituents to abstract
SOW Input
SBAR I will carry NP
If S I will VP
SOW
SOW Output
SBAR NP I carry
I will VP if S
Source reordering
SBAR I will carry NP
1 3 4 5 2
If S I will VP
4 5 1 2 3
REORDERPHRASE:
Use seq2seq model to reorder phrase

Figure 3: Overview of the source sentence rearrangement workflow for one level of recursion at the root node. First, candidate tree segment pairs contained within the input node are selected. A transduction operation is applied over the abstracted phrase, giving the reordering **4 5 1 2 3** for the case shown in red, then the process recursively continues for each abstracted node. This results in a reordering for the full source sentence; the reordering indices serve as additional input to the REAP model.

and N layers in the decoder. We make the target reordering $\mathbf{r}$ accessible to this transformer model through an additional set of positional embeddings $PE_{\mathbf{r}}$. We use the sinusoidal function to construct these following Vaswani et al. (2017).

Let $E_M = \text{encoder}_M(\mathbf{x})$ be the output of the M^{th} (last) layer of the encoder. The special-purpose position embeddings are added to the output of this layer (see Figure 2): $E = E_M + PE_r$. Note that these are separate from standard position embeddings added at the input layer; such embeddings are also used in our model to encode the original order of the source sentence. The transformer decoder model attends over E while computing attention and the presence of the position embeddings should encourage the generation to obey the desired ordering $\mathbf{r}$, while still conforming to the decoder language model. Our experiments in Section 4.3 show that this position embedding method is able to successfully guide the generation of paraphrases, conditioning on both the input sentence semantics as well as the desired ordering.

2.3 Sentence Reordering

We now outline our approach for generating these desired reorderings $\mathbf{r}$. We do this by predicting phrasal rearrangements with the SOW model at various levels of syntactic abstraction of the sentence. We combine multiple such phrase-level rearrangements to obtain a set R of sentence-level rearrangements. This is done using a top-down approach, starting at the root node of the parse tree. The overall recursive procedure is outlined in Algorithm 1.

Algorithm 1 REORDER(t)

```
Input: Sub-tree $t$ of the input parse tree
Output: Top-$k$ list of reorderings for $t$'s yield
  $\mathcal{T}$ = SELECTSEGMENTPAIRS($t$)              // Step 1
  $\mathcal{R}$ = INITIALIZEBEAM(size = $k$)
  for $(A, B)$ in $\mathcal{T}$ do
    $z$ = REORDERPHRASE($t, A, B$)                   // Step 2
    $R_A(1, \ldots, k)$ = REORDER($t_A$)            // $k$ orderings
    $R_B(1, \ldots, k)$ = REORDER($t_B$)            // $k$ orderings
    for $r_a, r_b$ in $R_A \times R_B$ do
      $r$ = COMBINE($z, r_a, r_b$)                    // Step 3
      score($r$) = score($z$)+score($r_a$)+score($r_b$)
      $\mathcal{R}$.push($r$, score($r$))
    end for
  end for
  return $\mathcal{R}$
```

One step of the recursive algorithm has three major steps: Figure 3 shows the overall workflow for one iteration (here, the root node of the sentence is selected for illustration). First, we select sub-phrase pairs of the input phrase that respect parse-tree boundaries, where each pair consists of non-overlapping phrases (Step 1). Since the aim is to learn generic syntax-governed rearrangements, we abstract out the two sub-phrases, and replace them with non-terminal symbols, retaining only the constituent tag information. For example, we show three phrase pairs in Figure 3 that can be abstracted away to yield the reduced forms of the sentences. We then use a seq2seq model to obtain rearrangements for each abstracted phrase (Step 2). Finally, this top-level rearrangement is combined with recursively-constructed phrase rearrangements within the abstracted phrases to obtain sentence-level rearrangements (Step 3).

Step 1: SELECTSEGMENTPAIRS

We begin by selecting phrase tuples that form the input to our seq2seq model. A phrase tuple (t, A, B) consists of a sub-tree t with the constituents A and B abstracted out (replaced by their syntactic categories). For instance, in Figure 3, the S_0, S, and VP_2 nodes circled in red form a phrase tuple. Multiple distinct combinations of A and B are possible.[2]

Step 2: REORDERPHRASE

Next, we obtain rearrangements for each phrase tuple (t, A, B). We first form an input consisting of the yield of t with A and B abstracted out; e.g. *If S I will VP*, shown in red in Figure 3. We use a sequence-to-sequence model (the SOW model) that takes this string as input and produces a corresponding output sequence. We then perform word-level alignment between the input and generated output sequences (using cosine similarity between GloVe embeddings) to obtain the rearrangement that must be applied to the input sequence.[3] The log probability of the output sequence serves as a score for this rearrangement.

SOW model The SOW model is a sequence-to-sequence model $P(\mathbf{y}' \mid \mathbf{x}', o)$, following the transformer framework in Section 2.1.[4] Both $\mathbf{x}'$ and $\mathbf{y}'$ are encoded using the word pieces vocabulary; additionally, embeddings corresponding to the POS tags and constituent labels (for non-terminals) are added to the input embeddings.

For instance, in Figure 3, *If S I will VP* and *I will VP if S* is an example of an $(\mathbf{x}', \mathbf{y}')$, pair. While not formally required, Algorithm 1 ensures that there are always exactly two non-terminal labels in these sequences. o is a variable that takes values MONOTONE or FLIP. This encodes a preference to keep the two abstracted nodes in the same order or to "flip" them in the output.[5] o is encoded in the model with additional positional encodings of the form $\{\ldots 0, 0, 1, 0, \ldots 2, 0 \ldots\}$ for monotone and $\{\ldots 0, 0, 2, 0, \ldots 1, 0 \ldots\}$ for flipped, wherein the non-zero positions correspond to the positions of the abstracted non-terminals in the phrase. These positional embeddings for the SOW MODEL are handled analogously to the $\mathbf{r}$ embeddings for the REAP model. During inference, we use both the monotone rearrangement and flip rearrangement to generate two reorderings, one of each type, for each phrase tuple.

We describe training of this model in Section 3.

Step 3: COMBINE

The previous step gives a rearrangement for the subtree t. To obtain a sentence-level rearrangement from this, we first recursively apply the REORDER algorithm on subtrees t_A and t_B which returns the top-k rearrangements of each subtree. We iterate over each rearrangement pair (r_a, r_b), applying these reorderings to the abstracted phrases A and B. This is illustrated on the left side of Figure 3. The sentence-level representations, thus obtained, are scored by taking a mean over all the phrase-level rearrangements involved.

3 Data and Training

We train and evaluate our model on the PARANMT-50M paraphrase dataset (Wieting and Gimpel, 2018) constructed by backtranslating the Czech sentences of the CzEng (Bojar et al., 2016) corpus. We filter this dataset to remove shorter sentences (less than 8 tokens), low quality paraphrase pairs (quantified by a translation score included with the dataset) and examples that exhibit low reordering (quantified by a reordering score based on the position of each word in the source and its aligned word in the target sentence). This leaves us with over 350k paired paraphrase pairs.

3.1 Training Data for REAP

To train our REAP model (outlined in Section 2.2), we take existing paraphrase pairs $(\mathbf{x}, \mathbf{y}^*)$ and derive pseudo-ground truth rearrangements $\mathbf{r}^*$ of the source sentence tokens based on their alignment with the target sentence. To obtain these rearrangements, we first get contextual embeddings (Devlin et al., 2019) for all tokens in the source and target sentences. We follow the strategy outlined in Lerner and Petrov (2013) and perform reorderings as we traverse down the dependency tree. Starting at the root node of the source sentence, we determine the order between the head and its children (independent of other decisions) based on the order

[2]In order to limit the number of such pairs, we employ a threshold on the fraction of non-abstracted words remaining in the phrase, outlined in more detail in the Appendix.

[3]We experimented with a pointer network to predict indices directly; however, the approach of generate and then align post hoc resulted in a much more stable model.

[4]See Appendix for SOW model architecture diagram.

[5]In syntactic translation systems, rules similarly can be divided by whether they preserve order or invert it (Wu, 1997).

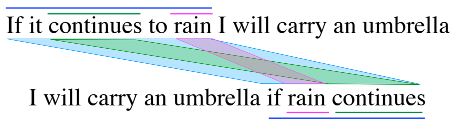
If it continues to rain I will carry an umbrella
I will carry an umbrella if rain continues

Figure 4: Paraphrase sentence pair and its aligned tuples $A \rightarrow B, C$ and $A' \rightarrow B', C'$. These produce the training data for the SOW MODEL.

of the corresponding aligned words in the target sentence. We continue this traversal recursively to get the sentence level-rearrangement. This mirrors the rearrangement strategy from Section 2.3, which operates over constituency parse tree instead of the dependency parse.

Given triples $(\mathbf{x}, \mathbf{r}^*, \mathbf{y}^*)$, we can train our REAP model to generate the final paraphrases conditioning on the pseudo-ground truth reorderings.

3.2 Training Data for SOW

The PARANMT-50M dataset contains sentence-level paraphrase pairs. However, in order to train our SOW model (outlined in section 2.3), we need to see phrase-level paraphrases with syntactic abstractions in them. We extract these from the PARANMT-50M dataset using the following procedure, shown in Figure 4. We follow Zhang et al. (2020) and compute a phrase alignment score between all pairs of constituents in a sentence and its paraphrase.[6] From this set of phrase alignment scores, we compute a partial one-to-one mapping between phrases (colored shapes in Figure 4); that is, not all phrases get aligned, but the subset that do are aligned one-to-one. Finally, we extract aligned chunks similar to rule alignment in syntactic translation (Galley et al., 2004): when aligned phrases A and A' subsume aligned phrase pairs (B, C) and (B', C') respectively, we can extract the aligned tuple (t_A, B, C) and $(t_{A'}, B', C')$. The phrases (B, C) and (B', C') are abstracted out to construct training data for the phrase-level transducer, including supervision of whether o = MONOTONE or FLIP. Using the above alignment strategy, we were able to obtain over 1 million aligned phrase pairs.

4 Evaluation

Setup As our main goal is to evaluate our model's ability to generate diverse paraphrases, we obtain a set of paraphrases and compare these to sets of paraphrases produced by other methods. To obtain 10 paraphrases, we first compute a set of 10 distinct reorderings $\mathbf{r}^1, \ldots, \mathbf{r}^{10}$ with the SOW method from Section 2.3 and then use the REAP to generate a 1-best paraphrase for each. We use top-k decoding to generate the final set of paraphrases corresponding to the reorderings. Our evaluation is done over 10k examples from PARANMT-50M.

4.1 Quantitative Evaluation

Baselines We compare our model against the Syntactically Controlled Paraphrase Network (**SCPN**) model proposed in prior work (Iyyer et al., 2018). It produces 10 distinct paraphrase outputs conditioned on a pre-enumerated list of syntactic templates. This approach has been shown to outperform other paraphrase approaches that condition on interpretable intermediate structures (Chen et al., 2019b). Additionally, we report results on the following baseline models: i) A **copy-input** model that outputs the input sentence exactly. ii) A vanilla **seq2seq** model that uses the same transformer encoder-decoder architecture from Section 2.1 but does not condition on any target rearrangement. We use top-k sampling (Fan et al., 2018) to generate 10 paraphrases from this model.[7] iii) A **diverse-decoding** model that uses the above transformer seq2seq model with diverse decoding (Kumar et al., 2019) during generation. Here, the induced diversity is uncontrolled and aimed at maximizing metrics such as distinct n-grams and edit distance between the generated sentences. iv) A **LSTM** version of our model where the REAP model uses LSTMs with attention (Bahdanau et al., 2014) and copy (See et al., 2017) instead of transformers. We still use the transformer-based phrase transducer to obtain the source sentence reorderings, and still use positional encodings in the LSTM attention.

Similar to Cho et al. (2019), we report two types of metrics:

1. **Quality**: Given k generated paraphrases $Y = \{\mathbf{y}^1, \mathbf{y}^2 \ldots \mathbf{y}^k\}$ for each input sentence in the test set, we select $\hat{\mathbf{y}}^{best}$ that achieves the best (oracle) sentence-level score with the ground truth paraphrase $\mathbf{y}$. The corpus level evaluation is performed using pairs $(\hat{\mathbf{y}}^{best}, \mathbf{y})$.
2. **Diversity**: We calculate BLEU or WER be-

[6]The score is computed using a weighted mean of the contextual similarity between individual words in the phrases, where the weights are determined by the corpus-level inverse-document frequency of the words. Details in the Appendix.

[7]Prior work (Wang et al., 2019; Li et al., 2019) has shown that such a transformer-based model provides a strong baseline and outperforms previous LSTM-based (Hasan et al., 2016) and VAE-based (Gupta et al., 2018) approaches.

Model	oracle quality (over 10 sentences, no rejection) ↑					pairwise diversity (post-rejection)	
	BLEU	ROUGE-1	ROUGE-2	ROUGE-L	% rejected	self-BLEU ↓	self-WER ↑
copy-input	18.4	54.4	27.2	49.2	0	–	–
SCPN	21.3	53.2	30.3	51.0	40.6	35.9	63.4
Transformer seq2seq	32.8	63.1	41.4	63.3	12.7	50.7	35.4
+ diverse-decoding	24.8	56.8	33.2	56.4	21.3	34.2	58.1
SOW-REAP (LSTM)	27.0	57.9	34.8	57.5	31.7	46.2	53.9
SOW-REAP	30.9	62.3	40.2	61.7	15.9	38.0	57.9

Table 1: Quality and diversity metrics for the different models. Our proposed approach outperforms other diverse models (SCPN and diverse decoding) in terms of all the quality metrics. These models exhibit higher diversity, but with many more rejected paraphrases, indicating that these models more freely generate bad paraphrases.

tween all pairs $(\mathbf{y}^i, \mathbf{y}^j)$ generated by a single model on a single sentence, then macro-average these values at a corpus-level.

In addition to these metrics, we use the paraphrase similarity model proposed by Wieting et al. (2017) to compute a paraphrase score for generated outputs with respect to the input. Similar to Iyyer et al. (2018), we use this score to filter out low quality paraphrases. We report on the rejection rate according to this criterion for all models. Note that our diversity metric is computed *after* filtering as it is easy to get high diversity by including nonsensical paraphrase candidates that differ semantically.

Table 1 outlines the performance of the different models. The results show that our proposed model substantially outperforms the SCPN model across all quality metrics.[8] Furthermore, our LSTM model also beats the performance of the SCPN model, demonstrating that the gain in quality cannot completely be attributed to the use of transformers. The quality of our full model (with rearrangements) is also comparable to the quality of the vanilla seq2seq model (without rearrangements). This demonstrates that the inclusion of rearrangements from the syntax-based neural transducer do not hurt quality, while leading to a substantially improved diversity performance.

The SCPN model has a high rejection score of 40.6%. This demonstrates that out of the 10 templates used to generate paraphrases for each sentence, on average 4 were not appropriate for the given sentence, and therefore get rejected. On the other hand, for our model, only 15.9% of the generated paraphrases get rejected, implying that the rearrangements produced were generally meaningful. This is comparable to the 12.7% rejection rate exhibited by the vanilla seq2seq model that does not condition on any syntax or rearrangement, and is therefore never obliged to conform to an inappropriate structure.

Finally, our model exhibits a much higher diversity within the generated paraphrases compared to the transformer seq2seq baseline. As expected, the SCPN model produces slightly more diverse paraphrases as it explicitly conditions the generations on templates with very different top level structures. However, this is often at the cost of semantic equivalence, as demonstrated by both quantitative and human evaluation (next section). A similar trend was observed with the diverse-decoding scheme. Although it leads to more diverse generations, there is a substantial decrease in quality compared to SOW-REAP and the seq2seq model. Moreover, the paraphrases have a higher rejection rate (21.3%), suggesting that diverse decoding is more likely to produce nonsensical paraphrases. A similar phenomenon is also reported by Iyyer et al. (2018), wherein diverse-decoding resulted in paraphrases with different semantics than the input.

Syntactic Exemplars In addition to SCPN, we compare our proposed model against the controllable generation method of Chen et al. (2019b). Their model uses an exemplar sentence as a syntactic guide during generation; the generated paraphrase is trained to incorporate the semantics of the input sentence while emulating the syntactic structure of the exemplar (see Appendix D for examples). However, their proposed approach depends on the availability of such exemplars at test time; they manually constructed these for their test set (800 examples). Since we do not have such example sentences available for our test data, we report results of our model's performance on their test data.

[8]The difference in performance between our proposed model and baseline models is statistically significant according to a paired bootstrap test.

Input	SOW-REAP	SCPN
if at any time in the preparation of this product the integrity of this container is compromised it should not be used .	this container should not be used if any time in the preparation of this product is compromised	in the preparation of this product , the integrity of this container is compromised , but it should not be used .
	if the integrity of the packaging is impaired at any time , the product should not be used .	where is the integrity of this product of this container the integrity of this container should not be used .
	if the product integrity of this container is compromised it should not be used .	i should not use if at any time in the preparation of this product , it should not be used .
i was the first grower to use hydroponics .	to use hydroponics , i was the first one .	where did i have the first tendency to use hydroponics ?
	i used hydroponics for the first time .	i used to use hydroponics .
	to use hydroponics the first time i was .	first i was the first grower to use hydroponics

Table 2: Examples of paraphrases generated by our system and the baseline SCPN model. Our model successfully rearranges the different structural components of the input sentence to obtain meaningful rearrangements. SCPN conforms to pre-enumerated templates that may not align with a given input.

Note that Chen et al. (2019b) carefully curated the exemplar to be syntactically similar to the actual target paraphrase. Therefore, for fair comparison, we report results using the ground truth ordering (that similarly leverages the target sentence to obtain a source reordering), followed by the REAP model. This model (ground truth order + REAP) achieves a 1-best BLEU score of 20.9, outperforming both the prior works: Chen et al. (2019b) (13.6 BLEU) and SCPN (17.8 BLEU with template, 19.2 BLEU with full parse). Furthermore, our full SOW-REAP model gets an oracle-BLEU (across 10 sentences) score of 23.8. These results show that our proposed formulation outperforms other controllable baselines, while being more flexible.

Input Sentence: if at any time in the preparation of this product the integrity of this container is compromised it should not be used .
Rule Sequence: if S it should not VB used . $\rightarrow$ should not VB used if S (parse tree level: 0)
at NP the integrity of this container VBZ compromised $\rightarrow$ this container VBZ weakened at NP (parse tree level: 1)
the NN of NP $\rightarrow$ NP NN (parse tree level: 2)
Generated Sentence: this container should not be used if the product is compromised at any time in preparation .

Table 3: Examples of our model's rearrangements applied to a given input sentence. Parse tree level indicates the rule subtree's depth from the root node of the sentence. The REAP model's final generation considers the rule reordering at the higher levels of the tree but ignores the rearrangement within the lower sub-tree.

4.2 Qualitative Evaluation

Table 2 provides examples of paraphrase outputs produced by our approach and SCPN. The examples show that our model exhibits syntactic diversity while producing reasonable paraphrases of the input sentence. On the other hand, SCPN tends to generate non-paraphrases in order to conform to a given template, which contributes to increased diversity but at the cost of semantic equivalence. In Table 3, we show the corresponding sequence of rules that apply to an input sentence, and the final generated output according to that input rearrangement. Note that for our model, on average, 1.8 phrase-level reorderings were combined to produce sentence-level reorderings (we restrict to a maximum of 3). More examples along with the input rule sequence (for our model) and syntactic templates (for SCPN) are provided in the Appendix.

Human Evaluation We also performed human evaluation on Amazon Mechanical Turk to evaluate the quality of the generated paraphrases. We randomly sampled 100 sentences from the development set. For each of these sentences, we obtained 3 generated paraphrases from each of the following models: i) SCPN, ii) vanilla sequence-to-sequence and iii) our proposed SOW-REAP model. We follow earlier work (Kok and Brockett, 2010; Iyyer et al., 2018) and obtain quality annotations on a 3 point scale: **0** denotes not a paraphrase, **1** denotes that the input sentence and the generated sentence are paraphrases, but the generated sentence might contain grammatical errors, **2** indicates that the input and the candidate are paraphrases. To emulate the human evaluation design in Iyyer et al. (2018), we sample paraphrases *after* filtering using the criterion outlined in the previous section and obtain three judgements per sentence and its 9 paraphrase candidates. Table 4 outlines the results from the human evaluation. As we can see, the results indicate

Model	2	1	0
SCPN (Iyyer et al., 2018)	35.9	24.8	39.3
Transformer seq2seq	45.1	20.6	34.3
SOW-REAP	44.5	22.6	32.9

Table 4: Human annotated quality across different models. The evaluation was done on a 3 point quality scale, 2 = grammatical paraphrase, 1 = ungrammatical paraphrase, 0 = not a paraphrase.

Ordering	oracle-ppl ↓	oracle-BLEU ↑
Monotone	10.59	27.98
Random	9.32	27.10
SOW	8.14	30.02
Ground Truth	7.79	36.40

Table 5: Comparison of different source reordering strategies. Our proposed approach outperforms baseline monotone and random rearrangement strategies.

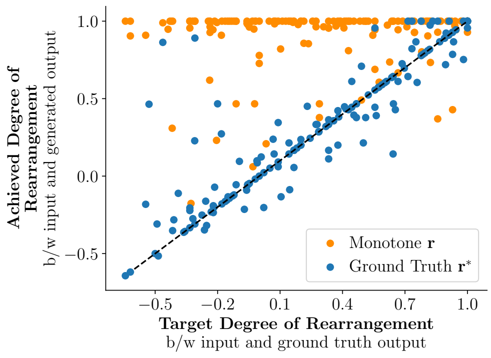

Figure 5: The degree of rearrangement (Kendall's Tau) achieved by conditioning on monotone and pseudo-ground truth reorderings ($\mathbf{r}^*$). The dotted line denotes the ideal performance (in terms of reordering-compliance) of the REAP model, when supplied with perfect reordering $\mathbf{r}^*$. The actual performance of the REAP model mirrors the ideal performance.

that the quality of the paraphrases generated from our model is substantially better than the SCPN model.[9] Furthermore, similar to quantitative evaluation, the human evaluation also demonstrates that the performance of this model is similar to that of the vanilla sequence-to-sequence model, indicating that the inclusion of target rearrangements do not hurt performance.

4.3 Ablations and Analysis

4.3.1 Evaluation of SOW Model

Next, we intrinsically evaluate the performance of our SOW model (Section 2.3). Specifically, given a budget of 10 reorderings, we want to understand how close our SOW model comes to covering the target ordering. We do this by evaluating the REAP model in terms of oracle perplexity (of the ground truth paraphrase) and oracle BLEU over these 10 orderings.

We evaluate our proposed approach against 3 systems: a) **Monotone** reordering $\{1, 2, \ldots, n\}$. b) **Random** permutation, by randomly permuting the children of each node as we traverse down the constituency parse tree. c) **Ground Truth**, using the pseudo-ground truth rearrangement (outlined in Section 3) between the source and ground-truth target sentence. This serves as an upper bound for the reorderings' performance, as obtained by the recursive phrase-level transducer.

[9]The difference of our model performance with SCPN is statistically significant, while that with baseline seq2seq is not according to a paired bootstrap test.

Table 5 outlines the results for 10 generated paraphrases from each rearrangement strategy. Our proposed approach outperforms the baseline monotone and random reordering strategies. Furthermore, the SOW model's oracle perplexity is close to that of the ground truth reordering's perplexity, showing that the proposed approach is capable of generating a diverse set of rearrangements such that one of them often comes close to the target rearrangement. The comparatively high performance of the ground truth reorderings demonstrates that the positional embeddings are effective at guiding the REAP model's generation.

4.3.2 Compliance with target reorderings

Finally, we evaluate whether the generated paraphrases follow the target reordering $\mathbf{r}$. Note that we do not expect or want our REAP model to be absolutely compliant with this input reordering since the model should be able to correct for the mistakes make by the SOW model and still generate valid paraphrases. Therefore, we perform reordering compliance experiments on only the monotone reordering and the pseudo-ground truth reorderings ($\mathbf{r}^*$, construction outlined in Section 3), since these certainly correspond to valid paraphrases.

For sentences in the test set, we generate paraphrases using monotone reordering and pseudo-ground truth reordering as inputs to REAP. We get the 1-best paraphrase and compute the degree of rearrangement[10] between the input sentence and

[10]Quantified by Kendall's Tau rank correlation between original source order and targeted/generated order. Higher

the generated sentence. In Figure 5, we plot this as a function of the target degree of rearrangement, i.e., the rearrangement between the input sentence $\mathbf{x}$ and the ground truth sentence $\mathbf{y}^*$. The dotted line denotes the ideal performance of the model in terms of agreement with the perfect reordering $\mathbf{r}^*$. The plot shows that the REAP model performs as desired; the monotone generation results in high Kendall's Tau between input and output. Conditioning on the pseudo-ground truth reorderings ($\mathbf{r}^*$) produces rearrangements that exhibit the same amount of reordering as the ideal rearrangement.

5 Related Work

Paraphrase Generation Compared to prior seq2seq approaches for paraphrasing (Hasan et al., 2016; Gupta et al., 2018; Li et al., 2018), our model is able to achieve much stronger controllability with an interpretable control mechanism. Like these approaches, we can leverage a wide variety of resources to train on, including backtranslation (Pavlick et al., 2015; Wieting and Gimpel, 2018; Hu et al., 2019) or other curated data sources (Fader et al., 2013; Lan et al., 2017).

Controlled Generation Recent work on controlled generation aims at controlling attributes such as sentiment (Shen et al., 2017), gender or political slant (Prabhumoye et al., 2018), topic (Wang et al., 2017), etc. However, these methods cannot achieve fine-grained control over a property like syntax. Prior work on diverse paraphrase generation can be divided into three groups: diverse decoding, latent variable modeling, and syntax-based. The first group uses heuristics such as Hamming distance or distinct n-grams to preserve diverse options during beam search decoding (Vijayakumar et al., 2018; Kumar et al., 2019). The second group includes approaches that use uninterpretable latent variables to separate syntax and semantics (Chen et al., 2019a), perturb latent representations to enforce diversity (Gupta et al., 2018; Park et al., 2019) or condition on latent codes used to represent different re-writing patterns (Xu et al., 2018; An and Liu, 2019). Qian et al. (2019) uses distinct generators to output diverse paraphrases. These methods achieve some diversity, but do not control generation in an interpretable manner. Finally, methods that use explicit syntactic structures (Iyyer et al., 2018; Chen et al., 2019b) may try to force a sentence to conform to unsuitable syntax. Phrase-level approaches (Li et al., 2019) are inherently less flexible than our approach.

Kendall's Tau indicates lower rearrangement and vice-versa.

Machine Translation Our work is inspired by pre-ordering literature in machine translation. These systems either use hand-crafted rules designed for specific languages (Collins et al., 2005; Wang et al., 2007) or automatically learn rewriting patterns based on syntax (Xia and McCord, 2004; Dyer and Resnik, 2010; Genzel, 2010; Khalilov and Simaan, 2011; Lerner and Petrov, 2013). There also exist approaches that do not rely on syntactic parsers, but induce hierarchical representations to leverage for pre-ordering (Tromble and Eisner, 2009; DeNero and Uszkoreit, 2011). In the context of translation, there is often a canonical reordering that should be applied to align better with the target language; for instance, head-final languages like Japanese exhibit highly regular syntax-governed reorderings compared to English. However, in diverse paraphrase generation, there doesn't exist a single canonical reordering, making our problem quite different.

In concurrent work, Chen et al. (2020) similarly use an additional set of position embeddings to guide the order of generated words for machine translation. This demonstrates that the REAP technique is effective for other tasks also. However, they do not tackle the problem of generating plausible reorderings and therefore their technique is less flexible than our full SOW-REAP model.

6 Conclusion

In this work, we propose a two-step framework for paraphrase generation: construction of diverse syntactic guides in the form of target reorderings followed by actual paraphrase generation that respects these reorderings. Our experiments show that this approach can be used to produce paraphrases that achieve a better quality-diversity trade-off compared to previous methods and strong baselines.

Acknowledgments

This work was partially supported by NSF Grant IIS-1814522, a gift from Arm, and an equipment grant from NVIDIA. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin for providing HPC resources used to conduct this research. Thanks as well to the anonymous reviewers for their helpful comments.

References

Zhecheng An and Sicong Liu. 2019. Towards Diverse Paraphrase Generation Using Multi-Class Wasserstein GAN. *arXiv preprint arXiv:1909.13827*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural Machine Translation by Jointly Learning to Align and Translate. *CoRR*, abs/1409.0473.

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

Ondřej Bojar, Ondřej Dušek, Tom Kocmi, Jindřich Libovickỳ, Michal Novák, Martin Popel, Roman Sudarikov, and Dušan Variš. 2016. CzEng 1.6: enlarged Czech-English parallel corpus with processing tools Dockered. In *International Conference on Text, Speech, and Dialogue*, pages 231–238. Springer.

Kehai Chen, Rui Wang, Masao Utiyama, and Eiichiro Sumita. 2020. Explicit Reordering for Neural Machine Translation. *arXiv preprint arXiv:2004.03818*.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019a. A Multi-Task Approach for Disentangling Syntax and Semantics in Sentence Representations. 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 2453–2464.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019b. Controllable Paraphrase Generation with a Syntactic Exemplar. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5972–5984, Florence, Italy. Association for Computational Linguistics.

Jaemin Cho, Minjoon Seo, and Hannaneh Hajishirzi. 2019. Mixture Content Selection for Diverse Sequence Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3112–3122.

Michael Collins, Philipp Koehn, and Ivona Kučerová. 2005. Clause restructuring for statistical machine translation. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 531–540. Association for Computational Linguistics.

John DeNero and Jakob Uszkoreit. 2011. Inducing sentence structure from parallel corpora for reordering. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 193–203. 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.

Chris Dyer and Philip Resnik. 2010. Context-free reordering, finite-state translation. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 858–866, Los Angeles, California. Association for Computational Linguistics.

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

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical Neural Story Generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898.

Michel Galley, Mark Hopkins, Kevin Knight, and Daniel Marcu. 2004. What's in a translation rule? In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics: HLT-NAACL 2004*, pages 273–280, Boston, Massachusetts, USA. Association for Computational Linguistics.

Dmitriy Genzel. 2010. Automatically learning source-side reordering rules for large scale machine translation. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 376–384.

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

Sadid A Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, Oladimeji Farri, et al. 2016. Neural Paraphrase Generation with Stacked Residual LSTM Networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934.

J. Edward Hu, Rachel Rudinger, Matt Post, and Benjamin Van Durme. 2019. ParaBank: Monolingual

Bitext Generation and Sentential Paraphrasing via Lexically-constrained Neural Machine Translation. In *Proceedings of AAAI*.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial Example Generation with Syntactically Controlled Paraphrase Networks. In *Proceedings of NAACL-HLT*, pages 1875–1885.

Maxim Khalilov and Khalil Simaan. 2011. Context-sensitive syntactic source-reordering by statistical transduction. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 38–46.

Stanley Kok and Chris Brockett. 2010. Hitting the right paraphrases in good time. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 145–153. Association for Computational Linguistics.

Ashutosh Kumar, Satwik Bhattamishra, Manik Bhandari, and Partha Talukdar. 2019. Submodular Optimization-based Diverse Paraphrasing and its Effectiveness in Data Augmentation. 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 3609–3619.

Wuwei Lan, Siyu Qiu, Hua He, and Wei Xu. 2017. A Continuously Growing Dataset of Sentential Paraphrases. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1224–1234, Copenhagen, Denmark. Association for Computational Linguistics.

Uri Lerner and Slav Petrov. 2013. Source-side classifier preordering for machine translation. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 513–523.

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, Brussels, Belgium. Association for Computational Linguistics.

Zichao Li, Xin Jiang, Lifeng Shang, and Qun Liu. 2019. Decomposable Neural Paraphrase Generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3403–3414, Florence, Italy. Association for Computational Linguistics.

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

Sunghyun Park, Seung-won Hwang, Fuxiang Chen, Jaegul Choo, Jung-Woo Ha, Sunghun Kim, and Jinyeong Yim. 2019. Paraphrase Diversification Using Counterfactual Debiasing. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 6883–6891.

Ellie Pavlick, Pushpendre Rastogi, Juri Ganitkevitch, Benjamin Van Durme, and Chris Callison-Burch. 2015. PPDB 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 425–430, Beijing, China. Association for Computational Linguistics.

Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of NAACL-HLT*, pages 2227–2237.

Shrimai Prabhumoye, Yulia Tsvetkov, Ruslan Salakhutdinov, and Alan W Black. 2018. Style Transfer Through Back-Translation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 866–876.

Lihua Qian, Lin Qiu, Weinan Zhang, Xin Jiang, and Yong Yu. 2019. Exploring Diverse Expressions for Paraphrase Generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3164–3173.

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.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725.

Tianxiao Shen, Tao Lei, Regina Barzilay, and Tommi Jaakkola. 2017. Style transfer from non-parallel text by cross-alignment. In *Advances in neural information processing systems*, pages 6830–6841.

Roy Tromble and Jason Eisner. 2009. Learning linear ordering problems for better translation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing: Volume 2-Volume 2*, pages 1007–1016. 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.

Ashwin K Vijayakumar, Michael Cogswell, Ramprasaath R Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. In *Thirty-Second AAAI Conference on Artificial Intelligence*.

Chao Wang, Michael Collins, and Philipp Koehn. 2007. Chinese syntactic reordering for statistical machine translation. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, pages 737–745.

Di Wang, Nebojsa Jojic, Chris Brockett, and Eric Nyberg. 2017. Steering Output Style and Topic in Neural Response Generation. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2140–2150.

Su Wang, Rahul Gupta, Nancy Chang, and Jason Baldridge. 2019. A task in a suit and a tie: paraphrase generation with semantic augmentation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 7176–7183.

John Wieting and Kevin Gimpel. 2018. ParaNMT-50M: Pushing the limits of paraphrastic sentence embeddings with millions of machine translations. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 451–462, Melbourne, Australia. Association for Computational Linguistics.

John Wieting, Jonathan Mallinson, and Kevin Gimpel. 2017. Learning Paraphrastic Sentence Embeddings from Back-Translated Bitext. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 274–285.

Dekai Wu. 1997. Stochastic inversion transduction grammars and bilingual parsing of parallel corpora. *Computational linguistics*, 23(3):377–403.

Fei Xia and Michael McCord. 2004. Improving a statistical MT system with automatically learned rewrite patterns. In *Proceedings of the 20th international conference on Computational Linguistics*, page 508. Association for Computational Linguistics.

Qiongkai Xu, Juyan Zhang, Lizhen Qu, Lexing Xie, and Richard Nock. 2018. D-PAGE: Diverse Paraphrase Generation. *CoRR*, abs/1808.04364.

Adams Wei Yu, David Dohan, Quoc Le, Thang Luong, Rui Zhao, and Kai Chen. 2018. QANet: Combining Local Convolution with Global Self-Attention for Reading Comprehension. In *International Conference on Learning Representations*.

Shiyue Zhang and Mohit Bansal. 2019. Addressing Semantic Drift in Question Generation for Semi-Supervised Question Answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2495–2509, Hong Kong, China. Association for Computational Linguistics.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2020. BERTScore: Evaluating Text Generation with BERT. In *International Conference on Learning Representations*.

Appendix

A SELECTSEGMENTPAIRS: Limiting number of segment pairs

As outlined in Section 2.3, the SELECTSEGMENTPAIRS subroutine returns a set of non-overlapping sub-phrases (A, B). In order to limit the number of sub-phrase pairs during inference, we employ the following heuristics:

1. We compute a score based on number of non-abstracted tokens divided by the total number of tokens in the yield of the parent sub-phrase t. We reject pairs (A, B) that have a score of more than 0.6. This reduces spurious ambiguity by encouraging the model to rearrange big constituents hierarchically rather than only abstracting out small pieces.
2. We maintain a list of tags that are never individually selected as sub-phrases. These include constituents that would be trivial to the reordering such as determiners (DT), prepositions (IN), cardinal numbers (CD), modals (MD), etc. However, these may be a part of larger constituents that form A or B.

B Training Data for SOW MODEL

In Section 3.2, we outlined our approach for obtaining phrase-level alignments from the PARANMT-50M dataset used to train the SOW MODEL. In the described approach, an alignment score is computed between each pair of phrases $p, \hat{p}$ belonging to sentences s and $\hat{s}$ respectively. We use the exact procedure in Zhang and Bansal (2019) to compute the alignment score, outlined below:

1. First, we compute an inverse document frequency (idf) score for each token in the training set. Let $M = \{s^{(i)}\}$ be the total number of sentences. Then idf of a word w is computed as:

$$idf(w) = -\log \frac{1}{M} \sum_{i=i}^{M} \mathbb{1}[w \in s^{(i)}]$$

2. Next, we extract a contextual representation of each word in the two phrases s and $\hat{s}$. We use ELMo (Peters et al., 2018) in our approach.

SOW Input	SOW Output
removing the NN from NP	excluding this NN from NP
they might consider VP if NP were imposed	in the case of imposition of NP , they would consider VP
NP lingered in the deserted NNS .	in the abandoned NNS , there was NP .
PP was a black NN archway .	was a black NN passage PP .
there is already a ring NN PP .	PP circular NN exist .

Table 6: Examples of aligned phrase pairs with exactly two sub-phrases abstracted out and replaced with constituent labels. These phrase pairs are used to train the SOW MODEL.

3. In order to compute a similarity score between each pair of phrases $(p, \hat{p})$, we use greedy matching to first align each token in the source phrase to its most similar word in the target phrase. To compute phrase-level similarity, these these word-level similarity scores are combined by taking a weighted mean, with weights specified by to the idf scores. Formally,

$$R_{p,\hat{p}} = \frac{\sum_{w_i \in p} idf(w_i) \max_{\hat{w}_j \in \hat{p}} w_i^T \hat{w}_j}{\sum_{w_i \in p} idf(w_i)}$$

$$P_{p,\hat{p}} = \frac{\sum_{\hat{w}_j \in \hat{p}} idf(\hat{w}_j) \max_{w_i \in p} w_i^T \hat{w}_j}{\sum_{\hat{w}_j \in \hat{p}} idf(\hat{w}_j)}$$

$$F_{p,\hat{p}} = \frac{2P_{p,\hat{p}} R_{p,\hat{p}}}{P_{p,\hat{p}} + R_{p,\hat{p}}}$$

This scoring procedure is exactly same as the one proposed by Zhang et al. (2020) to evaluate sentence and phrase similarities.

4. Finally, the phrases $p \in s$ and $\hat{p} \in \hat{s}$ are aligned if:

$$p = \underset{p_i \in s}{\operatorname{argmax}} F_{p_i,\hat{p}} \;\; \& \;\; \hat{p} = \underset{\hat{p_j} \in \hat{s}}{\operatorname{argmax}} F_{p,\hat{p_j}}$$

These aligned set of phrase pairs $(p, \hat{p})$ are used to construct tuples (t_A, B, C) and (t'_A, B', C'), as outlined in Section 3.2. Table 6 provides examples of such phrase pairs.

C SOW Model Architecture

Figure 6 provides an overview of the SOW seq2seq model. We add POS tag embeddings (or corresponding constituent label embeddings for abstracted X and Y) to the input token embeddings and original order position embeddings. As outlined in Section 2.3, another set of position embeddings corresponding to the order preference, either MONOTONE or FLIP, are further added to the output of the final layer of the encoder. The decoder attends over these augmented encodings during both training and inference.

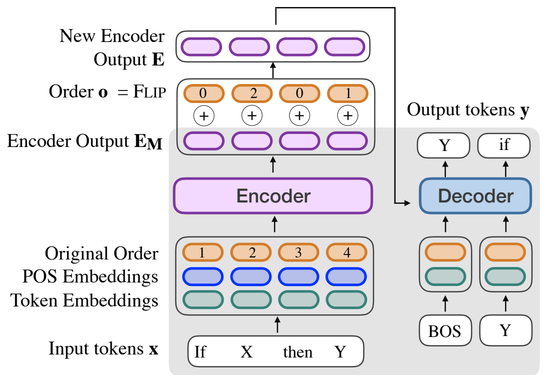

Figure 6: Source Order reWriting (SOW) model. Our model encodes order preference MONOTONE or FLIP through position embeddings added to the encoder output.

D Syntactic Exemplars

Table 7 provides an example from the test set of Chen et al. (2019b). The output retains the semantics of the input sentence while following the structure of the exemplar.

I:	his teammates eyes got an ugly, hostile expression.
E:	the smell of flowers was thick and sweet.
O:	the eyes of his teammates had turned ugly and hostile.

Table 7: Example of input (I), syntactic exemplar (E), and the reference output (O) from the evaluation test set of (Chen et al., 2019b).

E Example Generations

In Table 8, we provide examples of paraphrases generated by our system (SOW-REAP) and the baseline SCPN (Iyyer et al., 2018) system. We additionally include the phrase level transductions applied to obtain the sentence level reordering by our system (column 1) and the input template that the corresponding SCPN generation was conditioned on (Column 3).

Rules (SOW)	Output (REAP)	Template (SCPN)	Output (SCPN)
Input: the public tender result message normally contains the following information :			
NP normally contains the following NN: → the following NN usually contains in NP :	the following information shall normally be included in the public procurement report :	SBARQ (WHADVP SQ .)	where is the public procurement report report usually contains the following information .
NP normally VP : → usually VP , NP VBZ the following NN → the NN VBZ	normally the following information shall be included in the public procurement result report :	S (PP , NP VP .)	in the public competition , the report on competition contains the following information .
Input: the story of obi-wan kenobi ends here .			
NP VP . → VP is NP the NN of NP → NP NN .	end of the obi-wan kenobi story .	S (VP .)	tell the story of obi-wan kenobi .
the story PP NNS here . → there NNS a story PP .	here ends the story of obi-wan kenobi .	S (S , CC S .)	the story of obi-wan kenobi is here , and it ends here .
Input: i leased it before i knew where the money came from .			
i VBN it before i VP . → before i VP , i VBN it .	before i knew where the money came from , i rented it .	SBARQ (WHADVP SQ .)	where did you learn that it was the money ?
NP knew SBAR . → SBAR , S knew .	where the money came from , i lent it to me before i knew .	S (NP VP .)	i borrowed money before i knew where the money came from .
Input: priority actions should be more clearly specified in future reviews .			
NP should be more clearly specified PP . → PP , NP should be clearly specified .	in future reviews , priority measures should be more clearly specified .	S (S , CC S .)	priority actions should be more clearly specified in future reviews , and they should be informed .
ADVP VBN in future reviews → VBN in future reviews ADVP	priority measures should be specified in future reviews clearly .	SBARQ (WHADVP SQ .)	where should priority actions are more clearly specified in future reviews ?
Input: okay , well , tonight the occasion is calling .			
ADJP , S . → S , ADJP . well , NN the occasion VP → the occasion VP , NN	the occasion is calling today , okay ?	S (NP VP .)	the opportunity is calling .
ADJP , S . → S , ADJP . well , NP VBZ calling → VBZ calling NP	we 'll call it tonight , okay ?	S (ADVP NP VP .)	of course , the occasion is calling .
Input: a minor risk considering the number of telephones in new york .			
a JJ risk considering NP . → NP is a JJ risk . the NN of NP → NP NN	phones in new york are a minor risk considering .	SBARQ (WHADVP SQ .)	when do you consider the number of telephones in new york ?
NP_1 considering NP_2 . → considering NP_2 for NP_1 NN of NP → NP NN NP in JJ york → JJ york NP	in new york , the number of phones is a minor risk .	FRAG (SBAR) .	that minor risk is the number of telephones in new york .
Input: that dress gets me into anywhere i want .			
that S i VBP . → i VBP S .	i want that dress gets me into the place .	NP (NP .)	that dress gets me in there , i wish .
that S i VBP . → i VBP S . NN gets me PP → PP , NN gets me .	i want a dress in front of me .	S (VP .)	i want everywhere .

Table 8: Examples of paraphrases generated by our system and the baseline SCPN model. The outputs from our model successfully rearranges the different structural components of the input sentence to obtain meaningful rearrangements. SCPN on the other hand tends to conform to pre-specified templates that are often not aligned with a given input.

F Implementation Details

The hyperparameters values used in REAP (see Table 9) and SOW (see Table 10) models. Note that we do not use coverage loss for the SOW model.

Seq2seq transformer architecture	
Hidden size	256
Num layers	2
Num heads	8
Dropout	0.1
Training	
Optimizer	Adam, $\beta = (0.9, 0.999), \epsilon = 10^{-8}$
Learning rate	0.0001
Batch size	32
Epochs	50 (maximum)
Coverage loss coeff.	1 (first 10 epochs), 0.5 (10 - 20 epochs), 0 (rest)
Inference	
k in top-k	20
Beam Size	10

Table 9: Hyperparameters used in the implementation of the REAP model.

Seq2seq transformer architecture	
Hidden size	256
Num layers	2
Num heads	8
Dropout	0.1
Training	
Optimizer	Adam, $\beta = (0.9, 0.999), \epsilon = 10^{-8}$
Learning rate	0.0001
Batch size	32
Epochs	50 (maximum)
Recombination of rules/transductions	
Ignored tags	DT, IN, CD, MD, TO, PRP
Max. no. of rules	3

Table 10: Hyperparameters used in the implementation of the SOW model.