Syntax-Guided Controlled Generation of Paraphrases

Ashutosh Kumar[1] **Kabir Ahuja**[2*] **Raghuram Vadapalli**[3*] **Partha Talukdar**[1]

[1]Indian Institute of Science, Bangalore
[2]Microsoft Research, Bangalore
[3]Google, London
ashutosh@iisc.ac.in, kabirahuja2431@gmail.com
raghuram.4350@gmail.com, ppt@iisc.ac.in

Abstract

Given a sentence (e.g., *''I like mangoes''*) and a constraint (e.g., sentiment flip), the goal of controlled text generation is to produce a sentence that adapts the input sentence to meet the requirements of the constraint (e.g., *''I hate mangoes''*). Going beyond such simple constraints, recent work has started exploring the incorporation of complex syntactic-guidance as constraints in the task of controlled paraphrase generation. In these methods, syntactic-guidance is sourced from a separate exemplar sentence. However, these prior works have only utilized limited syntactic information available in the parse tree of the exemplar sentence. We address this limitation in the paper and propose **S**yntax **G**uided **C**ontrolled **P**araphraser (SGCP), an end-to-end framework for syntactic paraphrase generation. We find that SGCP can generate syntax-conforming sentences while not compromising on relevance. We perform extensive automated and human evaluations over multiple real-world English language datasets to demonstrate the efficacy of SGCP over state-of-the-art baselines. To drive future research, we have made SGCP's source code available.[1]

1 Introduction

Controlled text generation is the task of producing a sequence of coherent words based on given constraints. These constraints can range from simple attributes like tense, sentiment polarity, and word-reordering (Hu et al., 2017; Shen et al., 2017; Yang et al., 2018) to more complex syntactic information. For example, given a sentence *''The movie is awful!''* and a *simple* constraint like flip sentiment to positive, a controlled text generator is expected to produce the sentence *''The movie is fantastic!''*.

These constraints are important in not only providing information about *what to say* but also *how to say it*. Without any constraint, the ubiquitous sequence-to-sequence neural models often tend to produce degenerate outputs and favor generic utterances (Vinyals and Le, 2015; Li et al., 2016). Although simple attributes are helpful in addressing *what to say*, they provide very little information about *how to say it*. Syntactic control over generation helps in filling this gap by providing that missing information.

Incorporating complex syntactic information has shown promising results in neural machine translation (Stahlberg et al., 2016; Aharoni and Goldberg, 2017; Yang et al., 2019), data-to-text generation (Peng et al., 2019), abstractive text-summarization (Cao et al., 2018), and adversarial text generation (Iyyer et al., 2018). Additionally, recent work (Iyyer et al., 2018; Kumar et al., 2019) has shown that augmenting lexical and syntactical variations in the training set can help in building better performing and more robust models.

In this paper, we focus on the task of syntactically controlled paraphrase generation, that is, given an input sentence and a syntactic exemplar, produce a sentence that conforms to the syntax of the exemplar while retaining the meaning of the original input sentence. While syntactically controlled generation of paraphrases finds applications in multiple domains like data-augmentation and text passivization, we highlight its importance in the particular task of text simplification. As pointed out in Siddharthan (2014), depending on the literacy skill of an individual, certain syntactical forms of English

[*] This research was conducted during the authors internship at Indian Institute of Science.

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

Transactions of the Association for Computational Linguistics, vol. 8, pp. 330–345, 2020. https://doi.org/10.1162/tacl_a_00318
Action Editor: Asli Celikyilmaz. Submission batch: 12/2019; Revision batch: 2/2020; Published 6/2020.

sentences are easier to comprehend than others. As an example, consider the following two sentences:

S1 *Because it is raining today, you should carry an umbrella.*

S2 *You should carry an umbrella today, because it is raining.*

Connectives that permit pre-posed adverbial clauses have been found to be difficult for third to fifth grade readers, even when the order of mention coincides with the causal (and temporal) order (Anderson and Davison, 1986; Levy, 2003). Hence, they prefer sentence **S2**. However, various other studies (Clark and Clark, 1968; Katz and Brent, 1968; Irwin, 1980) have suggested that for older school children, college students, and adults, comprehension is better for the cause-effect presentation, hence sentence **S1**. Thus, modifying a sentence, syntactically, would help in better comprehension based on literacy skills.

Prior work in syntactically controlled paraphrase generation addressed this task by conditioning the semantic input on either the features learned from a linearized constituency-based parse tree (Iyyer et al., 2018), or the *latent* syntactic information (Chen et al., 2019a) learned from exemplars through variational auto-encoders. Linearizing parse trees typically results in loss of essential dependency information. On the other hand, as noted in Shi et al. (2016), an autoencoder-based approach might not offer rich enough syntactic information as guaranteed by actual constituency parse trees. Moreover, as noted in Chen et al. (2019a), SCPN (Iyyer et al., 2018), and CGEN (Chen et al., 2019a) tend to generate sentences of the same length as the exemplar. This is an undesirable characteristic because it often results in producing sentences that end abruptly, thereby compromising on grammaticality and semantics. Please see Table 1 for sample generations using each of the models.

To address these gaps, we propose Syntax Guided Controlled Paraphraser (SGCP) which uses *full* exemplar syntactic tree information. Additionally, our model provides an easy mechanism to incorporate different levels of syntactic control (granularity) based on the height of the tree being considered. The decoder in our framework is augmented with rich enough syntactical information to be able to produce syntax conforming sentences while not losing out on semantics and grammaticality.

SOURCE	*– how do i predict the stock market ?*
EXEMPLAR	*– can a brain transplant be done ?*
SCPN	*– how can the stock and start ?*
CGEN	*– can the stock market actually happen ?*
SGCP (Ours)	*– can i predict the stock market ?*
SOURCE	*– what are some of the mobile apps you ca n't live without and why ?*
EXEMPLAR	*– which is the best resume you have come across ?*
SCPN	*– what are the best ways to lose weight ?*
CGEN	*– which is the best mobile app you ca n't ?*
SGCP (Ours)	*– which is the best app you ca n't live without and why ?*

Table 1: Sample syntactic paraphrases generated by SCPN (Iyyer et al., 2018), CGEN (Chen et al., 2019a), SGCP (Ours). We observe that SGCP is able to generate syntax conforming paraphrases without compromising much on relevance.

The main contributions of this work are as follows:

1. We propose SGCP, an end-to-end model to generate syntactically controlled paraphrases at different levels of granularity using a parsed exemplar.

2. We provide a new decoding mechanism to incorporate syntactic information from the exemplar sentence's syntactic parse.

3. We provide a dataset formed from Quora Question Pairs[2] for evaluating the models. We also perform extensive experiments to demonstrate the efficacy of our model using multiple automated metrics as well as human evaluations.

2 Related Work

Controllable Text Generation. is an important problem in NLP that has received significant attention in recent times. Prior work include generating text using models conditioned on attributes like formality, sentiment, or tense (Hu et al., 2017; Shen et al., 2017; Yang et al., 2018) as well as on syntactical templates (Iyyer et al., 2018; Chen et al., 2019a). These systems find applications in adversarial sample generation (Iyyer et al., 2018),

[2] https://www.kaggle.com/c/quora-question-pairs.

text summarization, and table-to-text generation (Peng et al., 2019). While achieving state-of-the-art in their respective domains, these systems typically rely on a known finite set of attributes thereby making them quite restrictive in terms of the styles they can offer.

Paraphrase Generation. While generation of paraphrases has been addressed in the past using traditional methods (McKeown, 1983; Barzilay and Lee, 2003; Quirk et al., 2004; Hassan et al., 2007; Zhao et al., 2008; Madnani and Dorr, 2010; Wubben et al., 2010), they have recently been superseded by deep learning-based approaches (Prakash et al., 2016; Gupta et al., 2018; Li et al., 2019, 2018; Kumar et al., 2019). The primary task of all these methods (Prakash et al., 2016; Gupta et al., 2018; Li et al., 2018) is to generate the most semantically similar sentence and they typically rely on beam search to obtain any kind of lexical diversity. Kumar et al. (2019) try to tackle the problem of achieving lexical, and limited syntactical diversity using submodular optimization but do not provide any syntactic control over the type of utterance that might be desired. These methods are therefore restrictive in terms of the syntactical diversity that they can offer.

Controlled Paraphrase Generation. Our task is similar in spirit to Iyyer et al. (2018) and Chen et al. (2019a), which also deals with the task of syntactic paraphrase generation. However, the approach taken by them is different from ours in at least two aspects. Firstly, SCPN (Iyyer et al., 2018) uses an attention-based (Bahdanau et al., 2014) pointer-generator network (See et al., 2017) to encode input sentences and a linearized constituency tree to produce paraphrases. Because of the linearization of syntactic tree, considerable dependency-based information is generally lost. Our model, instead, directly encodes the tree structure to produce a paraphrase. Secondly, the inference (or generation) process in SCPN is computationally very expensive, because it involves a two-stage generation process. In the first stage, they generate full parse trees from incomplete templates, and then from full parse trees to final generations. In contrast, the inference in our method involves a single-stage process, wherein our model takes as input a semantic source, a syntactic tree and the level of syntactic style that needs to be transferred, to obtain the generations. Additionally, we also observed that the model does not perform well in low resource settings. This, again, can be attributed to the compounding implicit noise in the training due to linearized trees and generation of full linearized trees before obtaining the final paraphrases.

Chen et al. (2019a) propose a syntactic exemplar-based method for controlled paraphrase generation using an approach based on latent variable probabilistic modeling, neural variational inference, and multi-task learning. This, in principle, is very similar to Chen et al. (2019b). As opposed to our model, which provides different levels of syntactic control of the exemplar-based generation, this approach is restrictive in terms of the flexibility it can offer. Also, as noted in Shi et al. (2016), an autoencoder-based approach might not offer rich enough syntactic information as offered by actual constituency parse trees. Additionally, VAEs (Kingma and Welling, 2014) are generally unstable and harder to train (Bowman et al., 2016; Gupta et al., 2018) than seq2seq-based approaches.

3 SGCP: Proposed Method

In this section, we describe the inputs and various architectural components essential for building SGCP, an end-to-end trainable model. Our model, as shown in Figure 1, comprises a sentence encoder (3.2), syntactic tree encoder (3.3), and a syntactic-paraphrase-decoder (3.4).

3.1 Inputs

Given an input sentence X and a syntactic exemplar Y, our goal is to generate a sentence Z that conforms to the syntax of Y while retaining the meaning of X.

The semantic encoder (Section 3.2) works on sequence of input tokens, and the syntactic encoder (Section 3.3) operates on constituency-based parse trees. We parse the syntactic exemplar Y[3] to obtain its constituency-based parse tree. The leaf nodes of the constituency-based parse tree consists of token for the sentence Y. These tokens, in some sense, carry the semantic information of sentence Y, which we do not need for generating paraphrases. In order to prevent any *meaning*

[3]Obtained using the Stanford CoreNLP toolkit (Manning et al., 2014).

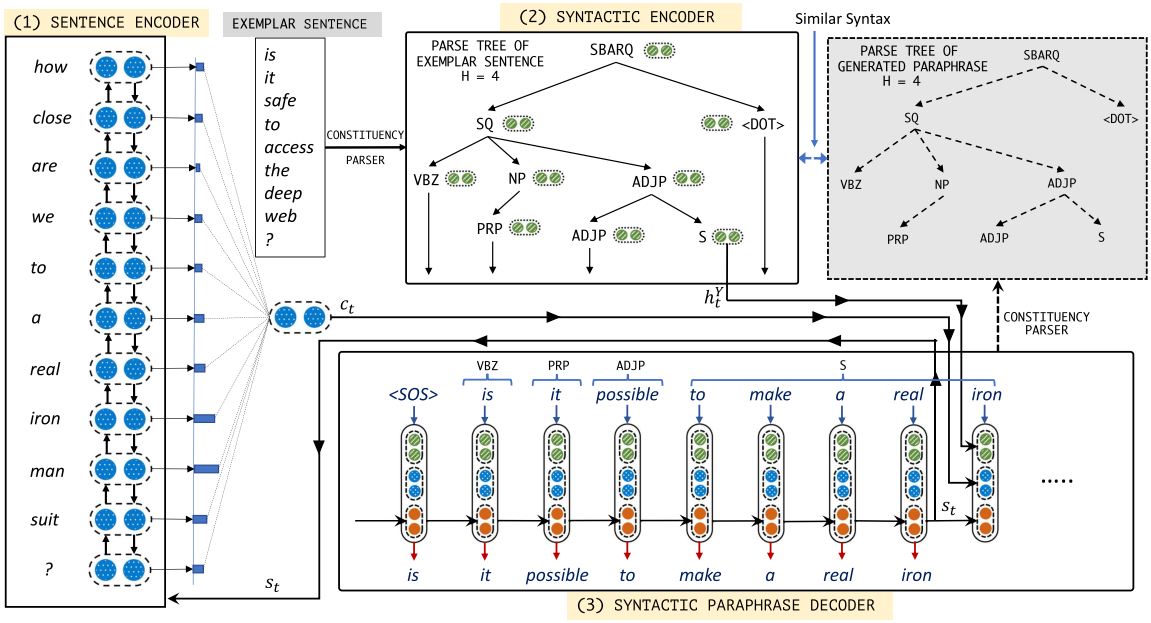


Figure 1: Architecture of SGCP (proposed method). SGCP aims to paraphrase an input sentence, while conforming to the syntax of an exemplar sentence (provided along with the input). The input sentence is encoded using the Sentence Encoder (Section 3.2) to obtain a semantic signal c_t. The Syntactic Encoder (Section 3.3) takes a constituency parse tree (pruned at height H) of the exemplar sentence as an input, and produces representations for all the nodes in the pruned tree. Once both of these are encoded, the Syntactic Paraphrase Decoder (Section 3.4) uses pointer-generator network, and at each time step takes the semantic signal c_t, the decoder recurrent state s_t, embedding of the previous token and syntactic signal h_t^Y to generate a new token. Note that the syntactic signal remains the same for each token in a span (shown in figure above curly braces; please see Figure 2 for more details). The gray shaded region (not part of the model) illustrates a qualitative comparison of the exemplar syntax tree and the syntax tree obtained from the generated paraphrase. Please refer to Section 3 for details.

propagation from exemplar sentence Y into the generation, we remove these leaf/terminal nodes from its constituency parse. The tree thus obtained is denoted as $\mathcal{C}^Y$.

The syntactic encoder, additionally, takes as input H, which governs the level of syntactic control needed to be induced. The utility of H will be described in Section 3.3.

3.2 Semantic Encoder

The semantic encoder, a multilayered Gated Recurrent Unit (GRU), receives tokenized sentence $X = \{x_1, \ldots, x_{T_X}\}$ as input and computes the contextualized hidden state representation h_t^X for each token using:

$$h_t^X = \text{GRU}(h_{t-1}^X, e(x_t)), \qquad (1)$$

where $e(x_t)$ represents the learnable embedding of the token x_t and $t \in \{1, \ldots, T_X\}$. Note that we use byte-pair encoding (Sennrich et al., 2016) for word/token segmentation.

3.3 Syntactic Encoder

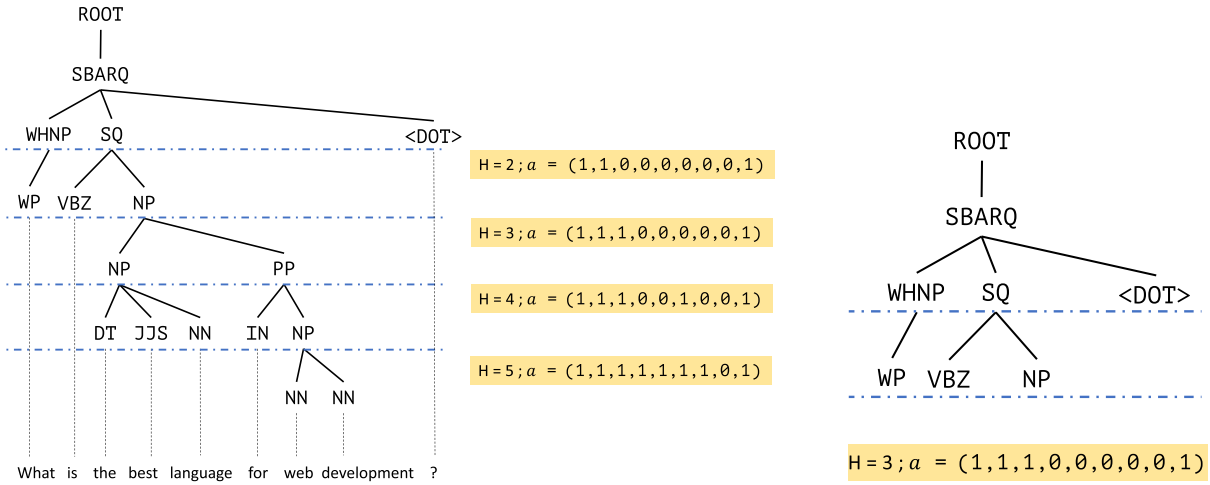
This encoder provides the necessary syntactic guidance for the generation of paraphrases. Formally, let constituency tree $\mathcal{C}^Y = \{\mathcal{V}, \mathcal{E}, \mathcal{Y}\}$, where $\mathcal{V}$ is the set of nodes, $\mathcal{E}$ the set of edges, and $\mathcal{Y}$ the labels associated with each node.

We calculate the hidden-state representation h_v^Y of each node $v \in \mathcal{V}$ using the hidden-state representation of its parent node $pa(v)$ and the embedding associated with its label y_v as follows:

$$h_v^Y = \text{GeLU}(W_{pa} h_{pa(v)}^Y + W_v e(y_v) + b_v), \qquad (2)$$

where $e(y_v)$ is the embedding of the node label y_v, and W_{pa}, W_v, b_v are learnable parameters. This approach can be considered similar to TreeLSTM (Tai et al., 2015). We use `GeLU` activation function (Hendrycks and Gimpel, 2016) rather than the standard `tanh` or `relu`, because of superior empirical performance.

As indicated in Section 3.1, syntactic encoder takes as input the height H, which governs the level of syntactic control. We randomly prune the



(a) Full Constituency Parse Tree

(b) Pruned tree at height H = 3

Figure 2: The constituency parse tree serves as an input to the syntactic encoder (Section 3.3). The first step is to remove the leaf nodes which contain *meaning representative tokens* (Here: What is the best language . . .). H denotes the height to which the tree can be pruned and is an input to the model. Figure 2(a) shows the full constituency parse tree annotated with vector $\boldsymbol{a}$ for different heights. Figure 2(b) shows the same tree pruned at height H = 3 with its corresponding $\boldsymbol{a}$ vector. The vector $\boldsymbol{a}$ serves as an *signalling* vector (Section 3.4.2) which helps in deciding the syntactic signal to be passed on to the decoder. Please refer Section 3 for details.

tree $\mathcal{C}^Y$ to height $H \in \{3, \ldots, H_{\text{max}}\}$, where H_{max} is the height of the full constituency tree $\mathcal{C}^Y$. As an example, in Figure 2b, we prune the constituency-based parse tree of the exemplar sentence, to height $H = 3$. The leaf nodes for this tree have the labels `WP`, `VBZ`, `NP`, and `<DOT>`. Although we calculate the hidden-state representation of all the nodes, only the terminal nodes are responsible for providing the syntactic signal to the decoder (Section 3.4).

We maintain a queue $\mathbb{L}_H^Y$ of such terminal node representations where elements are inserted from left to right for a given H. Specifically, for the particular example given in Figure 2b,

$$\mathbb{L}_H^Y = [h_{\text{WP}}^Y, h_{\text{VBZ}}^Y, h_{\text{NP}}^Y, h_{\text{<DOT>}}^Y]$$

We emphasize the fact that the length of the queue $|\mathbb{L}_H^Y|$ is a function of height H.

3.4 Syntactic Paraphrase Decoder

Having obtained the semantic and syntactic representations, the decoder is tasked with the generation of syntactic paraphrases. This can be modeled as finding the best $Z = Z^*$ that maximizes the probability $\mathbb{P}(Z|X, Y)$, which can further be factorized as:

$$Z^* = \arg\max_z \prod_{t=1}^{T_Z} (z_t | z_1, \ldots, z_{t-1}, X, Y), \quad (3)$$

where T_Z is the maximum length up to which decoding is required.

In the subsequent sections, we use t to denote the decoder time step.

3.4.1 Using Semantic Information

At each decoder time step t, the attention distribution α^t is calculated over the encoder hidden states h_i^X, obtained using Equation 1, as:

$$e_i^t = v^\intercal \tanh(W_h h_i^X + W_s s_t + b_{\text{attn}})$$
$$\alpha^t = \text{softmax}(e^t), \quad (4)$$

where s_t is the decoder cell-state and $v, W_h, W_s, b_{\text{attn}}$ are learnable parameters.

The attention distribution provides a way to jointly align and train sequence to sequence models by producing a weighted sum of the semantic encoder hidden states, known as context-vector c_t, given by:

$$c_t = \sum_i \alpha_i^t h_i^X \quad (5)$$

c_t serves as the semantic signal which is essential for generating meaning preserving sentences.

3.4.2 Using Syntactic Information

During *training*, each terminal node in the tree $\mathcal{C}^Y$, pruned at H, is equipped with information about the span of words it needs to generate. At each

time step t, only *one* terminal node representation $h_v^Y \in \mathbb{L}_H^Y$ is responsible for providing the syntactic signal which we call h_t^Y. This hidden-state representation to be used is governed through an *signalling* vector $\boldsymbol{a} = (a_1, \ldots, a_{T_z})$, where each $a_i \in \{0, 1\}$. 0 indicates that the decoder should keep on using the same hidden-representation $h_v^Y \in \mathbb{L}_H^Y$ that is currently being used, and 1 indicates that the next element (hidden-representation) in the queue $\mathbb{L}_H^Y$ should be used for decoding.

The utility of $\boldsymbol{a}$ can be best understood through Figure 2b. Consider the syntactic tree pruned at height $H = 3$. For this example,

$$\mathbb{L}_H^Y = [h_{\texttt{WP}}^Y, h_{\texttt{VBZ}}^Y, h_{\texttt{NP}}^Y, h_{\texttt{<DOT>}}^Y]$$

and

$$\boldsymbol{a} = (1, 1, 1, 0, 0, 0, 0, 0, 1)$$

$a_i = 1$ provides a signal to pop an element from the queue $\mathbb{L}_H^Y$ while $a_i = 0$ provides a signal to keep on using the last popped element. This element is then used to guide the decoder *syntactically* by providing a signal in the form of hidden-state representation (Equation 8).

Specifically, in this example, the $a_1 = 1$ signals $\mathbb{L}_H^Y$ to pop $h_{\texttt{WP}}^Y$ to provide syntactic guidance to the decoder for generating the first token. $a_2 = 1$ signals $\mathbb{L}_H^Y$ to pop $h_{\texttt{VBZ}}^Y$ to provide syntactic guidance to the decoder for generating the second token. $a_3 = 1$ helps in obtaining $h_{\texttt{NP}}^Y$ from $\mathbb{L}_H^Y$ to provide guidance to generate the third token. As described earlier, $a_4, \ldots, a_8 = 0$ indicates, that the same representation $h_{\texttt{NP}}^Y$ should be used for syntactically guiding tokens $z_4, \ldots, z_8$. Finally $a_9 = 1$ helps in retrieving $h_{\texttt{<DOT>}}^Y$ for guiding decoder to generate token z_9. Note that $|\mathbb{L}_H^Y| = \sum_{i=1}^{T_z} a_i$

Although $\boldsymbol{a}$ is provided to the model during training, this information might not be available during inference. Providing $\boldsymbol{a}$ during generation makes the model restrictive and might result in producing ungrammatical sentences. SGCP is tasked to learn a proxy for the *signalling* vector $\boldsymbol{a}$, using *transition probability vector* $\boldsymbol{p}$.

At each time step t, we calculate $p_t \in (0, 1)$, which determines the probability of changing the syntactic signal using:

$$p_t = \sigma(W_{\text{bop}}([c_t; h_t^Y; s_t; e(z_t')]) + b_{\text{bop}}), \quad (6)$$

$$h_{t+1}^Y = \begin{cases} h_t^Y & p_t < 0.5 \\ \texttt{pop}(\mathbb{L}_H^Y) & \text{otherwise} \end{cases} \quad (7)$$

where $\texttt{pop}$ removes and returns the next element in the queue, s_t is the decoder state, and $e(z_t')$ is the embedding of the input token at time t during decoding.

3.4.3 Overall

The semantic signal c_t, together with decoder state s_t, embedding of the input token $e(z_t')$ and the syntactic signal h_t^Y is fed through a GRU followed by softmax of the output to produce a vocabulary distribution as:

$$\mathbb{P}_{\text{vocab}} = \text{softmax}(W([c_t; h_t^Y; s_t; e(z_t')]) + b), \quad (8)$$

where $[;]$ represents concatenation of constituent elements, and W, b are trainable parameters.

We augment this with the copying mechanism (Vinyals et al., 2015) as in the pointer-generator network (See et al., 2017). Usage of such a mechanism offers a probability distribution over the extended vocabulary (the union of vocabulary words and words present in the source sentence) as follows:

$$\begin{aligned} \mathbb{P}(z) &= p_{\text{gen}} \mathbb{P}_{\text{vocab}}(z) + (1 - p_{\text{gen}}) \sum_{i:z_i=z} \alpha_i^t \\ p_{\text{gen}} &= \sigma(w_c^\intercal c_t + w_s^\intercal s_t + w_x^\intercal e(z_t') + b_{gen}) \end{aligned} \quad (9)$$

where w_c, w_s, w_x and b_{gen} are learnable parameters, $e(z_t')$ is the input token embedding to the decoder at time step t, and α_i^t is the element corresponding to the i^{th} co-ordinate in the attention distribution as defined in Equation 4.

The overall objective can be obtained by taking negative log-likelihood of the distributions obtained in Equation 6 and Equation 9.

$$\begin{aligned} \mathcal{L} = -\frac{1}{T} \sum_{t=0}^{T} [&\log \mathbb{P}(z_t^*) \\ &+ a_t \log(p_t) \\ &+ (1 - a_t) \log(1 - p_t)] \end{aligned} \quad (10)$$

where a_t is the t^{th} element of the vector $\boldsymbol{a}$.

4 Experiments

Our experiments are geared towards answering the following questions:

Q1. Is SGCP able to generate syntax conforming sentences without losing out on meaning? (Section 5.1, 5.4)

Q2. What level of syntactic control does SGCP offer? (Section 5.2, 5.3, 5.2)

Q3. How does SGCP compare against prior models, qualitatively? (Section 5.4)

Q4. Are the improvements achieved by SGCP statistically significant? (Section 5.1)

Based on these questions, we outline the methods compared (Section 4.1), along with the datasets (Section 4.2) used, evaluation criteria (Section 4.3) and the experimental setup (Section 4.4).

4.1 Methods Compared

As in Chen et al. (2019a), we first highlight the results of the two direct return-input baselines.

1. **Source-as-Output**: Baseline where the output is the semantic input.

2. **Exemplar-as-Output**: Baseline where the output is the syntactic exemplar.

We compare the following competitive methods:

3. **SCPN** (Iyyer et al., 2018) is a sequence-to-sequence based model comprising two encoders built with LSTM (Hochreiter and Schmidhuber, 1997) to encode semantics and syntax respectively. Once the encoding is obtained, it serves as an input to the LSTM-based decoder, which is augmented with soft-attention (Bahdanau et al., 2014) over encoded states as well as a copying mechanism (See et al., 2017) to deal with out-of-vocabulary tokens.[4]

4. **CGEN** (Chen et al., 2019a) is a VAE (Kingma and Welling, 2014) model with two encoders to project semantic input and syntactic input to a latent space. They obtain a syntactic embedding from one encoder, using a standard Gaussian prior. To obtain the semantic representation, they use von Mises-Fisher prior, which can be thought of as a Gaussian distribution on a hypersphere. They train the model using a multi-task paradigm, incorporating paraphrase generation loss and word position loss. We considered their best model, VGVAE + LC + WN + WPL, which incorporates the above objectives.

5. **SGCP (Section 3)** is a sequence-and-tree-to-sequence based model that encodes semantics and tree-level syntax to produce paraphrases. It uses a GRU-based (Chung et al., 2014) decoder with soft-attention on semantic encodings and a *begin of phrase* (bop) gate to select a leaf node in the exemplar syntax tree. We compare the following two variants of SGCP:

   (a) SGCP-F: Uses full constituency parse tree information of the exemplar for generating paraphrases.

   (a) SGCP-R: SGCP can produce multiple paraphrases by pruning the exemplar tree at various heights. This variant first generates five candidate generations, corresponding to five different heights of the exemplar tree, namely, $\{H_{\max}, H_{\max}-1, H_{\max}-2, H_{\max}-3, H_{\max}-4\}$, for each (source, exemplar) pair. From these candidates, the one with the highest ROUGE-1 score with the source sentence is selected as the final generation.

Note that, except for the return-input baselines, all methods use beam search during inference.

4.2 Datasets

We train the models and evaluate them on the following datasets:

(1) ParaNMT-small (Chen et al., 2019a) contains 500K sentence-paraphrase pairs for training, and 1,300 manually labeled sentence-exemplar-reference, which is further split into 800 test data points and 500 dev. data points, respectively.

As in Chen et al. (2019a), our model uses only (sentence, paraphrase) during training. The paraphrase itself serves as the exemplar input during training.

This dataset is a subset of the original ParaNMT-50M dataset (Wieting and Gimpel, 2018). ParaNMT-50M is a data set generated automatically through backtranslation of original English sentences. It is inherently noisy because of imperfect neural machine translation quality, with many sentences being non-grammatical and some even being non-English sentences. Because of such noisy data points, it is optimistic to assume that the corresponding constituency parse tree would be well aligned. To that end, we propose to

[4]Note that the results for SCPN differ from the ones shown in Iyyer et al. (2018). This is because the dataset used in Iyyer et al. (2018) is at least 50 times larger than the largest dataset (ParaNMT-small) in this work.

use the following additional dataset, which is more well-formed and has more human intervention than the ParaNMT-50M dataset.

(2) QQP-Pos: The original Quora Question Pairs (QQP) dataset contains about 400K sentence pairs labeled positive if they are duplicates of each other and negative otherwise. The dataset is composed of about 150K positive and 250K negative pairs. We select those positive pairs that contain both sentences with a maximum token length of 30, leaving us with ∼146K pairs. We call this dataset QQP-Pos.

Similar to ParaNMT-small, we use only the sentence-paraphrase pairs as training set and sentence-exemplar-reference triples for testing and validation. We randomly choose 140K sentence-paraphrase pairs as the training set $\mathbb{T}_{train}$, and the remaining 6K pairs $\mathbb{T}_{eval}$ are used to form the evaluation set $\mathbb{E}$. Additionally, let $\mathbb{T}_{eset} = \bigcup\{\{X, Z\} : (X, Z) \in \mathbb{T}_{eval}\}$. Note that $\mathbb{T}_{eset}$ is a set of sentences while $\mathbb{T}_{eval}$ is a set of sentence-paraphrase pairs.

Let $\mathbb{E} = \phi$ be the initial evaluation set. For selecting exemplar for each *each sentence-paraphrase pair* $(X, Z) \in \mathbb{T}_{eval}$, we adopt the following procedure:

Step 1: For a given $(X, Z) \in \mathbb{T}_{eval}$, construct an exemplar candidate set $\mathbb{C} = \mathbb{T}_{eset} - \{X, Z\}$. $|\mathbb{C}| \approx 12,000$.

Step 2: Retain only those sentences $C \in \mathbb{C}$ whose sentence length (= number of tokens) differ by at most two when compared to the paraphrase Z. This is done since sentences with similar constituency-based parse tree structures tend to have similar token lengths.

Step 3: Remove those candidates $C \in \mathbb{C}$, which are very similar to the source sentence X, that is, $\texttt{BLEU}(X, C) > 0.6$.

Step 4: From the remaining instances in $\mathbb{C}$, choose that sentence C as the exemplar Y which has the least Tree-Edit distance with the paraphrase Z of the selected pair, namely, $Y = \underset{C \in \mathbb{C}}{\text{argmin}}\ \texttt{TED}(Z, C)$. This ensures that the constituency-based parse tree of the exemplar Y is quite similar to that of Z, in terms of Tree-Edit distance.

Step 5: $\mathbb{E} := \mathbb{E} \cup (X, Y, Z)$.

Step 6: Repeat procedure for all other pairs in $\mathbb{T}_{eval}$.

From the obtained evaluation set $\mathbb{E}$, we randomly choose 3K triplets for the test set $\mathbb{T}_{test}$, and remaining 3K for the validation set $\mathbb{V}$.

4.3 Evaluation

It should be noted that there is no single fully reliable metric for evaluating syntactic paraphrase generation. Therefore, we evaluate on the following metrics to showcase the efficacy of syntactic paraphrasing models.

1. **Automated Evaluation**.

   (i) Alignment based metrics: We compute BLEU (Papineni et al., 2002), METEOR (Banerjee and Lavie, 2005), ROUGE-1, ROUGE-2, and ROUGE-L (Lin, 2004) scores between the generated and the reference paraphrases in the test set.

   (ii) Syntactic Transfer: We evaluate the syntactic transfer using Tree-edit distance (Zhang and Shasha, 1989) between the parse trees of:

   (a) the generated and the syntactic exemplar in the test set - **TED-E**
   (b) the generated and the reference paraphrase in the test set - **TED-R**

   (iii) Model-based evaluation: Because our goal is to generate paraphrases of the input sentences, we need some measure to determine if the generations indeed convey the same meaning as the original text. To achieve this, we adopt a model-based evaluation metric as used by Shen et al. (2017) for Text Style Transfer and Isola et al. (2017) for Image Transfer. Specifically, classifiers are trained on the task of Paraphrase Detection and then used as Oracles to evaluate the generations of our model and the baselines. We fine-tune two RoBERTa (Liu et al., 2019) based sentence pair classifiers, one on Quora Question Pairs (*Classifier-1*) and other on ParaNMT + PAWS[5] datasets (*Classifier-2*),

[5]Because the ParaNMT dataset only contains paraphrase pairs, we augment it with the PAWS (Zhang et al., 2019) dataset to acquire negative samples.

Model	BLEU↑	METEOR↑	ROUGE-1↑	ROUGE-2↑	ROUGE-L↑	TED-R↓	TED-E↓	PDS↑
QQP-Pos								
Source-as-Output	17.2	31.1	51.9	26.2	52.9	16.2	16.6	99.8
Exemplar-as-Output	16.8	17.6	38.2	20.5	43.2	4.8	0.0	10.7
SCPN (Iyyer et al., 2018)	15.6	19.6	40.6	20.5	44.6	9.1	8.0	27.0
CGEN (Chen et al., 2019a)	34.9	37.4	62.6	42.7	65.4	6.7	6.0	65.4
SGCP-F	**36.7**	**39.8**	**66.9**	**45.0**	**69.6**	**4.8**	**1.8**	**75.0**
SGCP-R	38.0	41.3	68.1	45.7	70.2	6.8	5.9	87.7
ParaNMT-small								
Source-as-Output	18.5	28.8	50.6	23.1	47.7	12.0	13.0	99.0
Exemplar-as-Output	3.3	12.1	24.4	7.5	29.1	5.9	0.0	14.0
SCPN (Iyyer et al., 2018)	6.4	14.6	30.3	11.2	34.6	6.2	**1.4**	15.4
CGEN (Chen et al., 2019a)	13.6	24.8	44.8	21.0	48.3	6.7	3.3	70.2
SGCP-F	**15.3**	**25.9**	**46.6**	21.8	**49.7**	**6.1**	**1.4**	**76.6**
SGCP-R	16.4	27.2	49.6	22.9	50.5	8.7	7.0	83.5

Table 2: Results on QQP and ParaNMT-small dataset. Higher↑ BLEU, METEOR, ROUGE, and PDS is better whereas lower↓ TED score is better. SGCP-R selects the best candidate out of many, resulting in performance boost for semantic preservation (shown in box). We bold the statistically significant results of SGCP-F, only, for a fair comparison with the baselines. Note that Source-as-Output, and Exemplar-as-Output are only dataset quality indicators and not the competitive baselines. Please see Section 5 for details.

which achieve accuracies of 90.2% and 94.0% on their respective test sets.[6]

Once trained, we use *Classifier-1* to evaluate generations on QQP-Pos and *Classifier-2* on ParaNMT-small.

We first generate syntactic paraphrases using all the models (Section 4.1) on the test splits of QQP-Pos and ParaNMT-small datasets. We then pair the source sentence with their corresponding generated paraphrases and send them as input to the classifiers. The Paraphrase Detection score, denoted as *PDS* in Table 2, is defined as, the ratio of the number of generations predicted as paraphrases of their corresponding source sentences by the classifier to the total number of generations.

2. **Human Evaluation.**
Although TED is sufficient to highlight syntactic transfer, there has been some scepticism regarding automated metrics for paraphrase quality (Reiter, 2018). To address this issue, we perform human evaluation on 100 randomly selected data points from the test set. In the evaluation, three judges (non-researchers proficient in the English language) were asked to assign scores to generated sentences based on the semantic similarity with the given source sentence. The annotators were shown a source sentence and the corresponding outputs of the systems in random order. The scores ranged from 1 (doesn't capture meaning at all) to 4 (perfectly captures the meaning of the source sentence).

[6]Because the test set of QQP is not public, the 90.2% number was computed on the available dev set (not used for model selection).

4.4 Setup

(a) Pre-processing. Because our model needs access to constituency parse trees, we tokenize and parse all our data points using the fully parallelizable Stanford CoreNLP Parser (Manning et al., 2014) to obtain their respective parse trees. This is done prior to training in order to prevent any additional computational costs that might be incurred because of repeated parsing of the same data points during different epochs.

(b) Implementation Details. We train both our models using the Adam Optimizer (Kingma and Ba, 2014) with an initial learning rate of 7e-5. We use a bidirectional three-layered GRU for encoding the tokenized semantic input and a standard pointer-generator network with GRU for decoding. The token embedding is learnable with dimension 300. To reduce the training complexity

Source	*what should be done to get rid of laziness ?*
Template Exemplar	*how can i manage my anger ?*
SCPN (Iyyer et al., 2018)	*how can i get rid ?*
CGEN (Chen et al., 2019a)	*how can i get rid of ?*
SGCP-F (Ours)	*how can i stop my laziness ?*
SGCP-R (Ours)	*how do i get rid of laziness ?*
Source	*what books should entrepreneurs read on entrepreneurship ?*
Template Exemplar	*what is the best programming language for beginners to learn ?*
SCPN (Iyyer et al., 2018)	*what are the best books books to read to read ?*
CGEN (Chen et al., 2019a)	*what 's the best book for entrepreneurs read to entrepreneurs ?*
SGCP-F (Ours)	*what is a best book idea that entrepreneurs to read ?*
SGCP-R (Ours)	*what is a good book that entrepreneurs should read ?*
Source	*how do i get on the board of directors of a non profit or a for profit organisation ?*
Template Exemplar	*what is the best way to travel around the world for free ?*
SCPN (Iyyer et al., 2018)	*what is the best way to prepare for a girl of a ?*
CGEN (Chen et al., 2019a)	*what is the best way to get a non profit on directors ?*
SGCP-F (Ours)	*what is the best way to get on the board of directors ?*
SGCP-R (Ours)	*what is the best way to get on the board of directors of a non profit or a for profit organisation ?*

Table 3: Sample generations of the competitive models. Please refer to Section 5.5 for details.

of the model, the maximum sequence length is kept at 60. The vocabulary size is kept at 24K for QQP and 50K for ParaNMT-small.

SGCP needs access to the level of syntactic granularity for decoding, depicted as H in Figure 2. During *training*, we keep on varying it randomly from 3 to $H_{\max}$, changing it with each training epoch. This ensures that our model is able to generalize because of an implicit regularization attained using this procedure. At each time-step of the decoding process, we keep a teacher forcing ratio of 0.9.

5 Results

5.1 Semantic Preservation and Syntactic Transfer

1. Automated Metrics: As can be observed in Table 2, our method(s) (SGCP-F/R (Section 4.1)) are able to outperform the existing baselines on both the datasets. Source-as-Output is independent of the exemplar sentence being used and since a sentence is a paraphrase of itself, the *paraphrastic scores* are generally high while the *syntactic scores* are below par. An opposite is true for Exemplar-as-Output. These baselines also serve as *dataset quality* indicators. It can be seen that source is semantically similar while being syntactically different from target sentence whereas the opposite is true when exemplar is compared to target sentences. Additionally, source sentences are syntactically and semantically different from exemplar sentences as can be observed from TED-E and PDS scores. This helps in showing that the dataset has rich enough syntactic diversity to learn from.

Through TED-E scores it can be seen that SGCP-F is able to adhere to the syntax of the exemplar template to a much larger degree than the baseline models. This verifies that our model is able to generate meaning preserving sentences while conforming to the syntax of the exemplars when measured using standard metrics.

It can also be seen that SGCP-R tends to perform better than SGCP-F in terms of *paraphrastic scores* while taking a hit on the *syntactic scores*. This makes sense, intuitively, because in some cases SGCP-R tends to select lower H values for syntactic granularity. This can also be observed from the example given in Table 6 where $H = 6$ is more favorable than $H = 7$, because of better meaning retention.

Although CGEN performs close to our model in terms of BLEU, ROUGE, and METEOR scores on ParaNMT-small dataset, its PDS is still much lower than that of our model, suggesting that our model is better at capturing the original meaning of the source sentence. In order to show that the results are not coincidental, we test the statistical significance of our model. We follow the non-parametric Pitman's permutation test (Dror et al., 2018) and observe that our model is statistically significant when the significance level (α) is taken to be 0.05. Note that this holds true for all metric on both the datasets except ROUGE-2 on ParaNMT-small.

	SCPN	CGEN	SGCP-F	SGCP-R
QQP-Pos	1.63	2.47	**2.70**	2.99
ParaNMT-small	1.24	1.89	**2.07**	2.26

Table 4: A comparison of human evaluation scores for comparing quality of paraphrases generated using all models. Higher score is better. Please refer to Section 5.1 for details.

2. Human Evaluation: Table 4 shows the results of human assessment. It can be seen that annotators, generally tend to rate SGCP-F and SGCP-R (Section 4.1) higher than the baseline models, thereby highlighting the efficacy of our models. This evaluation additionally shows that automated metrics are somewhat consistent with the human evaluation scores.

5.2 Syntactic Control

1. Syntactical Granularity: Our model can work with different levels of granularity for the exemplar syntax, namely, different tree heights of the exemplar tree can be used for decoding the output.

As can been seen in Table 6, at height 4 the syntax tree provided to the model is not enough to generate the full sentence that captures the meaning of the original sentence. As we increase the height to 5, it is able to capture the semantics better by predicting *some of* in the sentence. We see that at heights 6 and 7 SGCP is able to capture both semantics and syntax of the source and exemplar, respectively. However, as we provide the complete height of the tree (i.e., 7), it further tries to follow the syntactic input more closely leading to sacrifice in the overall relevance since the original sentence is about *pure substances* and not *a pure substance*. It can be inferred from this example that because a source sentence and exemplar's syntax might not be fully compatible with each other, using the complete syntax tree can potentially lead to loss of relevance and grammaticality. Hence by choosing different levels of syntactic granularity, one can address the issue of compatibility to a certain extent.

2. Syntactic Variety: Table 5 shows sample generations of our model on multiple exemplars for a given source sentence. It can be observed that SGCP can generate high-quality outputs for a variety of different template exemplars even the ones which differ a lot from the original sentence in terms of their syntax. A particularly interesting exemplar is *what is chromosomal mutation ? what are some examples ?*. Here, SGCP is able to generate a sentence with two question marks while preserving the essence of the source sentence. It should also be noted that the exemplars used in Table 5 were selected manually from the test sets, considering only their *qualitative compatibility* with the source sentence. Unlike the procedure used for the creation of QQP-Pos dataset, the final *paraphrases* were not kept in hand while selecting the *exemplars*. In real-world settings, where a *gold paraphrase* won't be present, these results are indicative of the qualitative efficacy of our method.

5.3 SGCP-R Analysis

ROUGE-based selection from the candidates favors paraphrases that have higher n-gram overlap with their respective source sentences, hence may capture source's meaning better. This hypothesis can be directly observed from the results in Tables 2 and 4, where we see higher values on automated semantic and human evaluation scores. Although this helps in obtaining better semantic generation, it tends to result in higher TED values. One possible reason is that, when provided with the complete tree, fine-grained information is available to the model for decoding and it forces the generations to adhere to the syntactic structure. In contrast, at lower heights, the model is provided with lesser syntactic information but equivalent semantic information.

5.4 Qualitative Analysis

As can be seen from Table 7, SGCP not only incorporates the best aspects of both the prior models, namely SCPN and CGEN, but also utilizes the complete syntactic information obtained using the constituency-based parse trees of the exemplar.

From the generations in Table 3, we can see that our model is able to capture both the semantics of the source text as well as the syntax of template. SCPN, evidently, can produce outputs with the template syntax, but it does so at the cost of semantics of the source sentence. This can also be verified from the results in Table 2, where SCPN performs poorly on PDS as compared with other models. In contrast, CGEN and SGCP retain much better semantic information, as is desirable. While generating sentences, CGEN often abruptly ends the

SOURCE: *how do i develop my career in software ?*	
SYNTACTIC EXEMPLAR	**SGCP GENERATIONS**
how can i get a domain for free ?	*how can i develop a career in software ?*
what is the best way to register a company ?	*what is the best way to develop career in software ?*
what are good places to visit in new york ?	*what are good ways to develop my career in software ?*
can i make 800,000 a month betting on horses ?	*can i develop my career in software ?*
what is chromosomal mutation ? what are some examples ?	*what is good career ? what are some of the ways to develop my career in software ?*
is delivery free on quikr ?	*is career useful in software ?*
is it possible to mute a question on quora ?	*is it possible to develop my career in software ?*

Table 5: Sample SGCP-R generations with a single source sentence and multiple syntactic exemplars. Please refer to Section 5.4 for details.

S	*what are pure substances ? what are some examples ?*
E	*what are the characteristics of the elizabethan theater ?*
H = 4	*what are pure substances ?*
H = 5	*what are some of pure substances ?*
H = 6	*what are some examples of pure substances ?*
H = 7	*what are some examples of a pure substance ?*

Table 6: Sample generations with different levels of syntactic control. S and E stand for source and exemplar, respectively. Please refer to Section 5.2 for details.

	Single-Pass	Syntactic Signal	Granularity
SCPN	✗	Linearized Tree	✓
CGEN	✓	POS Tags (During training)	✗
SGCP	✓	Constituency Parse Tree	✓

Table 7: Comparison of different syntactically controlled paraphrasing methods. Please refer to Section 5.4 for details.

sentence, as in example 1 in Table 3, truncating the penultimate token with *of*. The problem of abrupt ending due to insufficient syntactic input length was highlighted in Chen et al. (2019a) and we observe similar trends. SGCP, on the other hand, generates more relevant and grammatical sentences.

Based on empirical evidence, SGCP alleviates this shortcoming, possibly due to dynamic syntactic control and decoding. This can be seen in, for example, example 3 in Table 3 where CGEN truncates the sentence abruptly (penultimate token = *directors*) but SGCP is able to generate relevant sentence without compromising on grammaticality.

5.5 Limitations and Future Directions

All natural language English sentences cannot necessarily be converted to any desirable syntax. We note that SGCP does not take into account the compatibility of source sentence and template exemplars and can freely generate syntax conforming paraphrases. This, at times, leads to imperfect paraphrase conversion and nonsensical sentences like example 6 in Table 5 (*is career useful in software ?*). Identifying compatible exemplars is an important but separate task in itself, which we defer to future work.

Another important aspect is that the task of paraphrase generation is inherently domain agnostic. It is easy for humans to adapt to new domains for paraphrasing. However, because of the nature of the formulation of the problem in NLP, all the baselines, as well as our model(s), suffer from dataset bias and are not directly applicable to new domains. A prospective future direction can be to explore it from the lens of domain independence.

Analyzing the utility of controlled paraphrase generations for the task of data augmentation is another interesting possible direction.

6 Conclusion

In this paper we proposed SGCP, an end-to-end framework for the task of syntactically controlled paraphrase generation. SGCP generates paraphrase of an input sentence while conforming to the syntax of an exemplar sentence provided along with the input. SGCP comprises a GRU-based sentence encoder, a modified RNN-based tree encoder, and a pointer-generator–based novel decoder. In contrast to previous work

that focuses on a limited amount of syntactic control, our model can generate paraphrases at different levels of granularity of syntactic control without compromising on relevance. Through extensive evaluations on real-world datasets, we demonstrate SGCP's efficacy over state-of-the-art baselines.

We believe that the above approach can be useful for a variety of text generation tasks including syntactic exemplar-based abstractive summarization, text simplification and data-to-text generation.

Acknowledgments

This research is supported in part by the Ministry of Human Resource Development (Government of India). We thank the action editor Asli Celikyilmaz and the three anonymous reviewers for their helpful suggestions in preparing the manuscript. We also thank Chandrahas for his indispensable comments on earlier drafts of this paper.

References

Roee Aharoni and Yoav Goldberg. 2017. Towards string-to-tree neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 132–140. Vancouver, Canada. Association for Computational Linguistics.

Richard Chase Anderson and Alice Davison. 1986. Conceptual and empirical bases of readability formulas. *Center for the Study of Reading Technical Report; no. 392*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72.

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.

Samuel R. Bowman, Luke Vilnis, Oriol Vinyals, Andrew Dai, Rafal Jozefowicz, and Samy Bengio. 2016. Generating sentences from a continuous space. In *Proceedings of The 20th SIGNLL Conference on Computational Natural Language Learning*, pages 10–21, Berlin, Germany. Association for Computational Linguistics.

Ziqiang Cao, Wenjie Li, Sujian Li, and Furu Wei. 2018. Retrieve, rerank and rewrite: Soft template based neural summarization. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 152–161.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019a. 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.

Mingda Chen, Qingming Tang, Sam Wiseman, and Kevin Gimpel. 2019b. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS 2014 Workshop on Deep Learning, December 2014*.

Herbert H. Clark and Eve V. Clark. 1968. Semantic distinctions and memory for complex sentences. *Quarterly Journal of Experimental Psychology*, 20(2):129–138.

Rotem Dror, Gili Baumer, Segev Shlomov, and Roi Reichart. 2018. The hitchhiker's guide to testing statistical significance in natural language processing. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1383–1392, Melbourne, Australia. Association for Computational Linguistics.

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*.

Samer Hassan, Andras Csomai, Carmen Banea, Ravi Sinha, and Rada Mihalcea. 2007. Unt: Subfinder: Combining knowledge sources for automatic lexical substitution. In *Proceedings of the 4th International Workshop on Semantic Evaluations*, pages 410–413. Association for Computational Linguistics.

Dan Hendrycks and Kevin Gimpel. 2016. Gaussian error linear units (gelus). *arXiv preprint arXiv:1606.08415*.

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

Zhiting Hu, Zichao Yang, Xiaodan Liang, Ruslan Salakhutdinov, and Eric P Xing. 2017. Toward controlled generation of text. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1587–1596. JMLR.org.

Judith W. Irwin. 1980. The effects of explicitness and clause order on the comprehension of reversible causal relationships. *Reading Research Quarterly*, pages 477–488.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A. Efros. 2017. Image-to-image translation with conditional adversarial networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1125–1134.

Mohit Iyyer, John Wieting, Kevin Gimpel, and Luke Zettlemoyer. 2018. Adversarial example generation with syntactically controlled paraphrase networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1875–1885, New Orleans, Louisiana. Association for Computational Linguistics.

Evelyn Walker Katz and Sandor B. Brent. 1968. Understanding connectives. *Journal of Memory and Language*, 7(2):501.

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

Diederik P. Kingma and Max Welling. 2014. Auto-encoding variational bayes. *In Proceedings of ICLR*.

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, Minneapolis, Minnesota. Association for Computational Linguistics.

Elena T. Levy. 2003. The roots of coherence in discourse. *Human Development*, 46(4): 169–188.

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2016. A diversity-promoting objective function for neural conversation models. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 110–119.

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.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*. https://www.aclweb.org/anthology/W04-1013.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv: 1907.11692*.

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

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*, pages 55–60.

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: A method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Hao Peng, Ankur Parikh, Manaal Faruqui, Bhuwan Dhingra, and Dipanjan Das. 2019. Text generation with exemplar-based adaptive decoding. 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 2555–2565, Minneapolis, Minnesota. Association for Computational Linguistics.

Aaditya Prakash, Sadid A. Hasan, Kathy Lee, Vivek Datla, Ashequl Qadir, Joey Liu, and Oladimeji Farri. 2016. Neural paraphrase generation with stacked residual LSTM networks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 2923–2934, Osaka, Japan. The COLING 2016 Organizing Committee.

Chris Quirk, Chris Brockett, and William Dolan. 2004. Monolingual machine translation for paraphrase generation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing*, pages 142–149.

Ehud Reiter. 2018. A structured review of the validity of BLEU. *Computational Linguistics*, 44(3):393–401.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017, Jul. 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, Vancouver, Canada.

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, Berlin, Germany.

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.

Xing Shi, Inkit Padhi, and Kevin Knight. 2016. Does string-based neural MT learn source syntax? In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1526–1534, Austin, Texas. Association for Computational Linguistics.

Advaith Siddharthan. 2014. A survey of research on text simplification. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Felix Stahlberg, Eva Hasler, Aurelien Waite, and Bill Byrne. 2016. Syntactically guided neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for*

Computational Linguistics (Volume 2: Short Papers), pages 299–305.

Kai Sheng Tai, Richard Socher, and Christopher D. Manning. 2015. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 1556–1566.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. In *Advances in Neural Information Processing Systems*, pages 2692–2700.

Oriol Vinyals and Quoc Le. 2015. A neural conversational model. *arXiv preprint arXiv: 1506.05869*.

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.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2010. Paraphrase generation as monolingual translation: Data and evaluation. In *Proceedings of the 6th International Natural Language Generation Conference*, pages 203–207. Association for Computational Linguistics.

Xuewen Yang, Yingru Liu, Dongliang Xie, Xin Wang, and Niranjan Balasubramanian. 2019. Latent part-of-speech sequences for neural machine translation.

Zichao Yang, Zhiting Hu, Chris Dyer, Eric P. Xing, and Taylor Berg-Kirkpatrick. 2018. Unsupervised text style transfer using language models as discriminators. In *Advances in Neural Information Processing Systems*, pages 7287–7298.

Kaizhong Zhang and Dennis Shasha. 1989. Simple fast algorithms for the editing distance between trees and related problems. *SIAM Journal on Computing*, 18(6):1245–1262.

Yuan Zhang, Jason Baldridge, and Luheng He. 2019. PAWS: Paraphrase adversaries from word scrambling. 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 1298–1308.

Shiqi Zhao, Cheng Niu, Ming Zhou, Ting Liu, and Sheng Li. 2008. Combining multiple resources to improve SMT-based paraphrasing model. *Proceedings of ACL-08: HLT*. pages 1021–1029.