# Syntactically-Constrained Paraphrase Generation with Tree-Adjoining Grammar

Sarah J. Widder

Advised by Robert Frank

April 15, 2020

# Abstract

The paraphrasing task is twofold: a good paraphrase must both maintain semantic similarity to the original sentence while also incorporating an interesting level of syntactic diversity. In this work, we propose a modified syntactically controlled paraphrase network (SCPN) that is trained to produce a paraphrase of the input sentence with a desired syntax, encoded as a sequence of Tree Adjoining Grammar elementary trees. First, we demonstrate that an SCPN using elementary trees can successfully produce paraphrases that follow the syntactic constraint, remain faithful to the key semantic content of the original sentences, and offer some level of syntactic diversity relative to the original sentences. We also show how sequence-to-sequence translation of elementary trees can be used to create an end-to-end paraphrase generation system that does not require the elementary tree sequence of a desired paraphrase. Next, we explore how this model architecture can be modified to simulate sequential human sentence planning. We compare how the modified model performs when provided with elementary trees in either linear or hierarchical order. The success of smaller models trained on hierarchically-ordered data gives some support to models of human sentence production that prioritize selection of hierarchically-relevant information.

# Contents

# 1 Introduction

## 1.1 Paraphrase Generation

Paraphrasing presents an interesting dilemma in natural language processing. The paraphrasing task is twofold: a good paraphrase must both maintain *semantic similarity* to the original sentence while also incorporating an interesting level of *syntactic diversity*. In other words, the paraphrase must mean largely the same thing as the reference sentence while using different structure and word choice. While this process is fairly intuitive to a human, the open-ended nature of paraphrase generation can present a computational challenge to paraphrasing models. There can be many possible paraphrases of a given sentence, and they may vary greatly in quality. It is difficult when training a model to produce paraphrases of sentences to ensure that the paraphrases are structurally diverse from the input sentences, while maintaining the important semantic content. In spite of the challenges involved, quality paraphrase generation systems have many possible uses in consumer applications such as dialogue systems as well as components in larger models for downstream tasks.

## 1.2 Previous Approaches

Paraphrase generation has become a popular but challenging task in natural language processing in recent years. Often, development of paraphrase generation systems is done in the process of improving model performance on downstream tasks. Paraphrase generation can aid in dataset augmentation or provide adversarial examples that can increase model robustness on other tasks.

Many recent approaches focus on leveraging existing neural machine translation methods to produce or identify paraphrases. Two large corpora of paraphrase pairs, ParaNMT-50M (Wieting and Gimpel, 2018) and ParaBank (Hu et al., 2019), were generated through back-translation of the Czech half of an English-Czech parallel corpus of movie subtitles. These approaches take advantage of large existing parallel corpora and pre-trained neural machine translation systems to produce large corpora of English paraphrase pairs. A similar neural approach treats paraphrases as a foreign language for translation (Zhou et al., 2019). However, translation and paraphrase generation are similar but not identical tasks. The resulting paraphrases from this approach can lack diversity. Mao and Lee (2019) demonstrate that simply parroting source sentences can outperform state-of-the-art models on standard metrics. While a 'good'

paraphrase is difficult to quantify, standard metrics do not seem to capture the ability of these models to produce novel, interesting paraphrases.

Beyond methods that leverage neural machine translation techniques, transformers and variational auto-encoders have also had promising results. Li et al. (2019) propose a decomposable, transformer-based paraphrase generation system that implements word, phrasal, and sentential paraphrase capabilities. Roy and Grangier (2019) highlight the contrast between neural machine translation approaches and human paraphrase generation, which does not incorporate translation as an intermediate step. They propose an unsupervised model that leverages a variational auto-encoder for monolingual paraphrasing. Adversarial training has also been used for paraphrase generation without additional linguistic information (Yang et al., 2019).

Numerous approaches to paraphrase generation incorporate lexical or syntactic constraints to control and direct paraphrase generation. A *lexical constraint* is a requirement on the output of a model to include or exclude particular lexical items. The development of ParaBank (Hu et al., 2019) leveraged lexical constraints to produce multiple paraphrases for each source sentence. Kajiwara (2019) uses negative lexical constraints to prevent key words from the input from appearing in the output, forcing rewriting to generate more diverse paraphrases. Qian et al. (2019) use reinforcement learning with multiple discriminators and generators to produce a diverse set of paraphrases for each source sentence. Each generator explores a novel way to paraphrase an input sentence, while one of the discriminators determines whether each output conveys the same meaning as the input. Gan and Ng (2019) approach question paraphrasing using phrasal paraphrase suggestion to trigger generation in a particular direction. A *syntactic constraint* is a requirement on the output of a model to conform to a particular grammatical structure. Chen et al. (2019) encode syntactic constraints using an unrelated sentence as a syntactic exemplar, training a model that uses the semantic content of one sentence and the syntactic structure of another to generate a syntactically interesting paraphrase.

A particularly promising approach to addressing the issue of ensuring syntactically diverse paraphrases comes from Iyyer et al. (2018), who propose *syntactically controlled paraphrase networks* (SCPNs) that are trained to produce a paraphrase of the input sentence with a desired syntax. The desired structure of the paraphrase is given as an extra input to a neural encoder-decoder model. The syntactic constraint was encoded as a *constituency parse*, in which a sentence is broken down into sub-phrases called *constituents*. The constituency

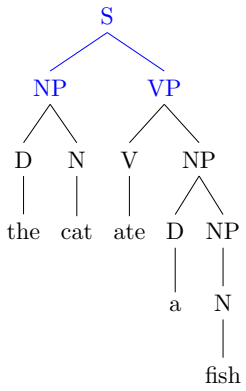


Figure 1: Constituency parse tree for *the cat ate a fish*, with the template highlighted in blue.

parses used were generated by the Stanford parser (Manning et al., 2014). The desired output structure was represented at varying levels of granularity; the input could be a simple template consisting of only the top two levels of the constituency parse or a full, linearized bracket parse without leaf nodes (i.e., tokens). For example, the full parse for *the cat ate a fish* might be `(S (NP (D) (N)) (VP (V) (NP (D) (N))))`, while the template would simply be `(S (NP) (VP))`. The bracketed constituency parse can also be represented with a tree as in Figure 1, where the nodes retained in the template are shown in blue.

In Iyyer et al. (2018), the template method was preferred over the unwieldy and overly-specific full parses, but also had significant drawbacks. When producing multiple paraphrases of an input sentence, the most frequent templates across the entire training corpus were used as syntactic inputs, regardless of the structure or content of the input sentence. The syntactic manipulation was generally successful in producing sentences that followed the desired structure, but the context-independent selection of templates led to sometimes nonsensical outputs or invalid paraphrases as the semantic content of the input sentence was forced into an unsuitable syntactic form. One such example given by the authors occurs when the model is given the sentence "with the help of captain picard , the borg will be prepared for everything ." and the template `(FRAG (INTJ) (,) (S) (,) (NP))`, and produces the output "oh , come on captain picard , the borg line for everything ." While this follows the syntactic structure of the template, the semantic content of the original sentence is not maintained.

A different representation of syntactic structure may be more well suited to

syntactically-controlled paraphrase generation. This work proposes a modification to the SCPN of Iyyer et al. (2018) that uses a sequence of elementary trees from Tree Adjoining Grammar (TAG) to represent the desired structure of the paraphrase instead of a constituency parse.

### 1.3 Tree Adjoining Grammar

Tree Adjoining Grammar (TAG) (Joshi et al., 1975) is a lexicalized grammar formalism that generates hierarchical structure through a system of tree rewriting. During TAG derivation, each word in a sentence is associated with an *elementary tree*, a piece of syntactic structure that encodes the syntactic constraints imposed by the word on the sentence. Each elementary tree encodes information about the structural positions of the word's dependents in addition to the dependencies headed by a word. For example, as shown in Figure 2, a transitive verb such as *ate* might be associated with the elementary tree t27, while a noun like *cat* or *fish* would be associated with the elementary tree t3. In these elementary trees, the nodes labeled with the diamond indicate the

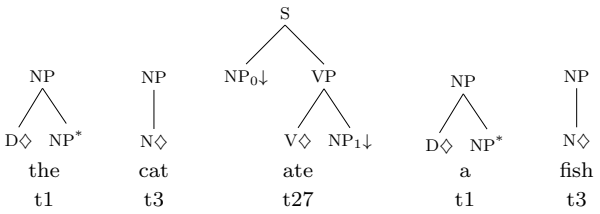


Figure 2: Elementary trees for *The cat ate a fish.*

structural position of the head of the tree. For the verbally-headed tree, the NP nodes that appear along the tree's frontier are the positions for the verb's arguments, i.e., its syntactic dependents. The subscripts on these arguments encode the deep syntactic relations with respect to the elementary tree's head (0 is subject, 1 is direct object, 2 is indirect object).

Two derivational operations, *substitution* and *adjoining* can be used to combine TAG elementary trees. Adjoining involves a special recursive elementary tree called an *auxiliary tree*, which has a *foot node* (marked with an asterisk) of the same category as the root node. For example, in Figure 2, tree t1 is an NP-recursive tree associated with a determiner such as *the*. During adjoining, a node N of category C in some elementary tree is rewritten with a C-recursive auxiliary tree T, and N's children are attached at the foot node of T. Thus, the

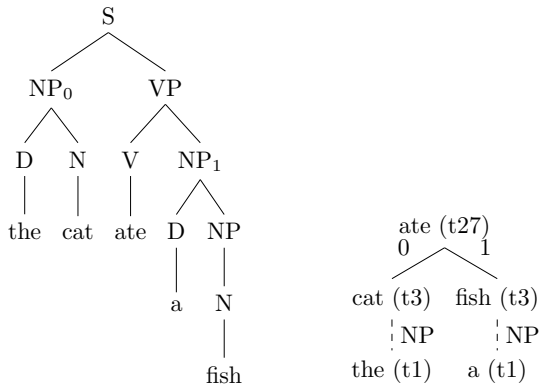


Figure 3: Derived and derivation trees for *the cat ate a fish*.

determiner tree t1 in Figure 2 can adjoin to the NP root of the N-headed tree t3, forming an NP-rooted tree corresponding to *the cat*.

In substitution, an elementary tree with a root node of category C is inserted into a leaf node of category C marked with a down arrow in another elementary tree. In the sentence *The cat ate a fish*, once the the auxiliary tree for *the* adjoins to the elementary tree for *cat*, the resulting structure can be substituted into the $NP_0$ substitution node in the S-rooted tree t27 shown in Figure 2. The results of these operations are shown in Figure 3: a *derived tree* (left), which shows the syntactic structure of the fully parsed sentence, and a *derivation tree* (right), which records the sequence of operations involved in the creation of the derived structure. The nodes of the derivation tree correspond to elementary trees and the edges (dependencies) correspond to substitution and adjoining operations that have applied, i.e., a child node is an elementary tree that has been substituted or adjoined into the parent node. Substitution is represented with solid lines and are labeled with the index of the substitution site (0 for subject and 1 for object), while adjoining is represented with dotted lines and labeled with the locus of adjoining.

The sequence of elementary trees that would be associated with a potential paraphrase of an input sentence can be used as the syntactic constraint in a paraphrase generation model similar to that of Iyyer et al. (2018). A sequence of elementary trees is almost a parse, as Bangalore and Joshi (1999) point out, and provides a relatively rich representation of the syntactic constraints on a sentence. For example, consider the passive sentence *a fish was eaten by the cat*, where the desired paraphrase might be the active form *the cat ate a fish*.

The constituency parse template for the desired paraphrase would be `(S (NP) (VP))`, while the sequence of elementary trees associated with each word of the desired paraphrase would be `t1 t3 t27 t1 t3` as in Figure 2. While the parse template allows for a greater variety of possible outputs, these may not always be interesting paraphrases of the source sentence. For example, the top two levels of the constituency parse of the input sentence *a fish was eaten by the cat* would yield the exact same parse template, meaning that the model could simply parrot the input while obeying the syntactic constraint. In contrast, the elementary tree sequence allows some flexibility in the output – the particular nouns and verbs chosen for each tree can vary – but the predicate elementary tree t27 ensures that the sentence must have an active verb with the agent on the left and the object on the right. This prevents the model from parroting the input and forces a syntactically diverse output that is still appropriate given the content of the original sentence.

In addition to providing a linguistically-rich representation of the desired syntactic structure of a paraphrase, using elementary trees rather than constituency parses also allows for more flexible manipulation of the syntactic constraint. In particular, we can manipulate the order in which elementary trees are presented to the decoder. The performance of the model under different ordering paradigms may give some insight into how humans plan and produce sentences, as discussed in the following sections.

## 1.4 Human Sentence Production

The paraphrasing task has many similarities to the general problem of human sentence production. To produce a sentence, a human must take some mental representation of the desired meaning of a sentence and generate an appropriately structured sequence of words to convey this meaning. The syntactic planning involved in this process is difficult to analyze. While it is unlikely a speaker has planned out the entire structure of their utterance and populated it with the appropriate lexical items before they begin speaking, it also clear that some planning must be done in advance. Influential models of sentence production (Bock and Levelt, 1994; Kempen and Hoenkamp, 1987; Levelt, 1989) highlight the importance of verbs in structural encoding, predicting that verb selection must occur early in sentence formulation. However, experimental tests have had mixed results: Schriefers et al. (1998) found no evidence for advance verb selection in verb-final utterances in German, while Momma et al. (2016) found

evidence that verbs are planned in advance in some constructions in Japanese, which is also verb-final. How exactly humans conduct syntactic planning, and in what order, remains unclear.

### 1.5 Current Work

The model presented here will follow the general architecture of the SCPN but will use sequences of Tree Adjoining Grammar (TAG) elementary trees rather than constituency parses to represent syntactic structure, allowing more linguistic richness and flexibility in how this information is provided to the model.[1] First, Section 2 presents the new SCPN using TAG elementary trees, and discusses model performance on a simplified dataset generated for the task. Next, Section 3 demonstrates how the model can be modified to simulate different paradigms of human sentence planning. The order of elementary trees can be manipulated to simulate linear sentence planning (choosing lexical items in left-to-right sentential order) or more hierarchically-driven sentence planning (choosing the verb and grammatical arguments first) that more closely matches the sentence production models discussed in Section 1.4. The performance of the SCPN given these two kinds of structural encoding may then give some insight into how syntactic planning is done in human sentence production. Section 4 will consider the significance of the results of the previous two sections, and finally Section 5 covers some future applications of this work.

## 2 Syntactically-Controlled Paraphrase Generation

### 2.1 Model Architecture

Figure 4 shows the model architecture of the syntactically constrained paraphrase network using elementary trees. Elementary trees are often referred to as supertags in neural applications; we will use these terms interchangeably. As with many neural approaches in language processing, recurrent neural networks are used for both encoders and the decoder. In a recurrent neural network, the input is fed to the network one token at a time, and each hidden unit is connected to the hidden unit at the previous step. This allows the network to learn a representation of a variable length input sequence as the computation unfolds

[1] Code and data available at https://github.com/sarah-widder/paraphrase

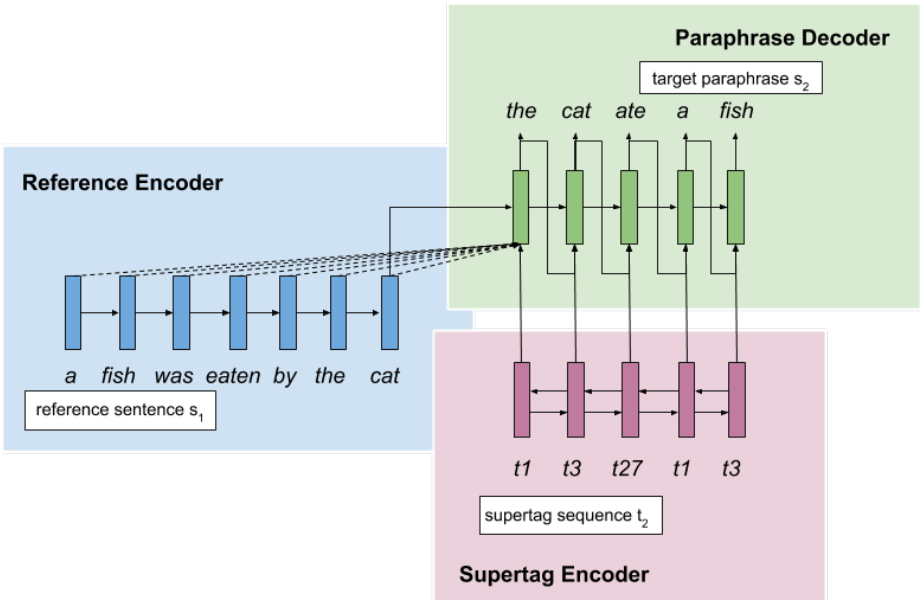


Figure 4: Model architecture. The reference sentence is encoded with a single-layer GRU, the supertag sequence is encoded with a bidirectional LSTM, and the decoder is a single-layer GRU. Dashed lines indicate attention over the reference sentence hidden states.

over time. In particular, this model uses a simple one-layer Gated Recurrent Unit (GRU, Cho et al. (2014)) for encoding the input sequence and decoding the output. Similar to the Long Short-Term Memory (LSTM, Hochreiter and Schmidhuber (1997)), a GRU includes gating mechanisms that control and manage the information flow between cells in a recurrent network. By regulating information flow with these gating mechanisms, a GRU or LSTM can adaptively capture dependencies across long sequences of data. The equations for the GRU are given in Figure 5.

Given a sentential paraphrase pair $\langle s_1, s_2 \rangle$ and a corresponding sequence of elementary trees $t_2$ for $s_2$, we encode $s_1$ with a unidirectional one-layer GRU. The sequence of elementary trees is encoded with a bidirectional LSTM. The equations for an LSTM are given in Figure 6. A bidirectional LSTM consists of one LSTM network that reads the elementary tree sequence from left to right, and one LSTM network that reads the elementary tree sequence from right to left, and the hidden states of each LSTM are concatenated. The output of the bidirectional LSTM is computed from the concatenation of the hidden states.

$$z_t = \sigma(W_z x_t + U_z h_{t-1} + b_z) \tag{1}$$

$$r_t = \sigma(W_r x_t + U_r h_{t-1} + b_r) \tag{2}$$

$$\tilde{h}_t = tanh(W_h x_t + U_h(r_t \odot h_{t-1}) + b_h) \tag{3}$$

$$h_t = z_t \odot h_{t-1} + (1 - z_t) \odot \tilde{h}_t \tag{4}$$

Figure 5: Equations for the GRU, where $x_t$ is the input vector, $h_t$ is the output vector, $z_t$ is the update gate vector, $r_t$ is the reset gate vector, and $W$, $U$, and $b$ are the parameter matrices and vector. $\sigma$ represents the activation function, either sigmoid or hyperbolic tangent.

$$f_t = \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \tag{5}$$

$$i_t = \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \tag{6}$$

$$o_t = \sigma_g(W_o x_t + U_o h_{t-1} + b_o) \tag{7}$$

$$\tilde{c}_t = \sigma_h(W_c x_t + U_c h_{t-1} + b_c) \tag{8}$$

$$c_t = f_t \circ c_{t-1} + i_t \circ \tilde{c}_t \tag{9}$$

$$h_t = o_t \circ \sigma_h(c_t) \tag{10}$$

Figure 6: Equations for the LSTM, where $x_t$ is the input vector, $f_t$ is the forget gate activation vector, $i_t$ is the update gate activation vector, $o_t$ is the output gate activation vector, $h_t$ is the hidden state vector, $\tilde{c}_t$ is the cell input activation vector, and $c_t$ is the cell state vector. $W$, $U$, and $b$ are the parameter matrices and vector. $\sigma$ is the activation function, either sigmoid or hyperbolic tangent.

The output of the bidirectional LSTM at each time step can thus get information from both the left context and the right context simultaneously. This ensures that the encoding of the syntactic constraint on the output takes into account the full sequence of elementary trees. This is particularly important in the case of paraphrasing active and passive sentences, as a unidirectional left-to-right encoding may not provide key information about the main verb of the output sentence early enough. For example, if the input is *a fish was eaten by the cat* and the sequence of elementary trees for the desired paraphrase is `t1 t3 t27 t1 t3` (as in Figure 2), the model must first choose a noun to output for *t3*, perhaps choosing *fish*. The model then receives the active verb form *t27*, produces *ate*, and has already failed to maintain a key piece of semantic information from the input sentence.

The decoder is a one-layer GRU augmented with attention over the encoded states (Bahdanau et al., 2015). In recurrent encoder-decoder models, *attention* is a weighted sum over the encoded states so that the decoder focuses on only the relevant aspects of the input sequence at each decoding step. In this case, at every time step the decoder receives the previous word (from the output or ground-truth) $w_{t-1}$, the attention-weighted average of encoder hidden states $a_t$, and the output of the bidirectional elementary tree encoder $z_t$, and computes

$$h_t = \text{GRU}[w_{t-1}; a_t; z_t]. \tag{11}$$

The decoder then computes a linear map of the hidden state followed by a softmax to get a probability distribution over the words in the vocabulary, and outputs the most likely word.

## 2.2 Data Generation

Two recent popular datasets of paraphrase pairs leverage parallel corpora and neural machine translation. Wieting and Gimpel (2018) started with a large Czech-English parallel corpus and used neural machine translation to translate the Czech sentences back to English, producing a set of 50 million paraphrase pairs, called PARANMT-50M. Hu et al. (2019) expand on this approach with lexical constraints that produce multiple sentential paraphrases for each source sentence in the same Czech-English corpus, called PARABANK. While these datasets offer a very large set of potential training pairs, the unconstrained nature of the machine translation method leads to quality concerns. The source material for the Czech-English parallel corpus is movie subtitles, which includes many sentence fragments and colloquialisms. Other issues include pairs of nearly-identical source and paraphrase sentences and paraphrases that no longer represent the original meaning of the source sentence. Table 1 gives some examples of problematic paraphrase pairs.

To simplify the task of the paraphrasing system and ensure control over the quality of paraphrase pairs, we generated training and test data using a simple rule-based system. Source sentences were generated with a lexicalized Context-Free Grammar (CFG) written in Python. The lexicon consisted of a limited set of agent nouns, location nouns, adjectives, prepositions, and transitive verbs, with a total vocabulary size of 100 tokens. Paraphrases were generated using a limited set of operations: active-to-passive and passive-to-active, modal substitution, synonym substitution, and prepositional phrase deletion and movement.

| Reference | Paraphrase |
|---|---|
| i 'm on a lasagna diet . | i 'm on a boring diet . |
| so , what 's your sign ? | so , what 's your signature ? " stop . " |
| sorry if i seemed doubtful in rome . | i 'm sorry i doubted rome . |
| you share a toothbrush ? ! | do you have a toothbrush ? |
| i don't want to come home . | i don't want to go home . |

Table 1: Examples of problematic paraphrase pairs from ParaNMT. Some paraphrases introduce new information that was not included in the reference sentence, while others lose or change information from the reference sentence. Some paraphrases are nearly identical to the source sentence.

| Paraphrase Operation | Reference | Paraphrase |
|---|---|---|
| Active/Passive | a bear near the house bit a boy . | a boy was bitten by a bear near the house . |
| Modal Substitution | a sleepy rabbit must hate a silly sheep . | a sleepy rabbit ought to hate a silly sheep . |
| Move PP | by the barn , a happy rabbit loved the cat . | a happy rabbit loved the cat by the barn . |

Table 2: Examples of paraphrase pairs generated using the partially-lexicalized CFG.

Examples are given in Table 2. The active-to-passive operation changes the voice of the sentence from active to passive, while the passive-to-active operation performs the same transformation in reverse. The modal substitution operation changes one form of a modal such as *must* to another, such as *ought to*. These modals can have different associated elementary trees, providing some diversity to the syntactic constraint. The prepositional phrase deletion operation simply removes a prepositional phrase that appeared in the reference sentence from the paraphrase sentence. The prepositional phrase movement operation either moves a prepositional phrase from the end of a sentence to the front, or from the beginning of a sentence to the end. For synonym substitution, we randomly select a noun or adjective from the paraphrase and replace it with a common synonym from WordNet (Fellbaum, 1998).

Each reference and paraphrase sentence was assigned part-of-speech tags using the Natural Language Toolkit POS tagger (Loper and Bird, 2002). Then, each sentence and the corresponding sequence of part of speech tags was passed to a pre-trained TAG supertagger (Kasai et al., 2017) that assigned a TAG elementary tree to each word in the sentence. These pairs of sentences $\langle s_1, s_2 \rangle$ and the sequence of supertags $t_2$ formed the inputs to the model. The training set consists of 171,452 paraphrase pairs, and the test set consists of 42,727 paraphrase pairs.

## 2.3 Evaluation Metrics

While the model is trained to produce only one paraphrase for each source sentence, there are many possible appropriate paraphrases for a given source sentence. These paraphrases may vary in both *semantic similarity* and *syntactic diversity* relative to the source sentence.

As a baseline assessment of each model's ability to produce the desired paraphrase, basic word-for-word (WFW) and supertag-for-supertag (SFS) accuracy measures are computed relative to the gold paraphrases in the evaluation set. The word-for-word accuracy is the percentage of correct words in the model output relative to the words in the test set paraphrases, allowing synonym substitution. The model outputs are also part-of-speech-tagged and fed into the same pre-trained supertagger, and these supertags are used to calculate supertag-for-supertag accuracy. This metric indicates how well the model learned to produce sentences that followed the syntactic constraints given in the form of the paraphrase supertag sequences. Additionally, the word-for-word accuracy is computed, allowing synonyms, among only the output words that had the correct supertag (word-for-supertag, WFS).

For a more linguistically informed measure of each model's ability to generate quality paraphrases (which may or may not match the desired paraphrase word for word), we use two other metrics for semantic fidelity and syntactic diversity. As an estimate of the semantic fidelity of the model output, we first generate a TAG parse of the output using the MICA parser (Bangalore et al., 2009), and compare the dependency relations in the output to those in the parses of the reference sentences. Dependency relations are trimmed to Root (main verb), 0 (subject), 1 (object), and each output sentence is scored on whether it maintains the original root, subject, and object separately. These relations capture the core meaning of these declarative transitive sentences - at the bare minimum, a good paraphrase must preserve the correct main verb and its subject and object. Each model is scored with the proportion of model outputs that maintained the root, subject, and object from the inputs.

As an estimate of syntactic diversity, we use ROUGE-L (Lin, 2004; Lin and Och, 2004), which identifies longest co-occurring in sequence n-grams between two sentences. A higher value in ROUGE-L indicates more similar sequences, so lower values are preferred in syntactically varied paraphrases. We report precision, recall, and f-score for this metric.

| Model | WFW | SFS | WFS |
|---|---|---|---|
| Hidden size: 50 | 71.3 | 93.8 | 73.9 |
| Hidden size: 100 | 80.1 | 94.2 | 82.3 |
| Hidden size: 256 | 97.1 | 97.8 | 97.4 |

Table 3: Word-for-word (WFW), supertag-for-supertag (SFS), and word-for-supertag (WFS) accuracy results for models trained with each hidden size.

| | ROUGE-L | | | TAG Dependency Overlap | | |
|---|---|---|---|---|---|---|
| Model | Precision | Recall | F-Score | Root | Subject | Object |
| Gold paraphrases | 0.594 | 0.534 | 0.553 | 0.981 | 0.936 | 0.963 |
| Hidden size: 50 | 0.465 | 0.386 | 0.412 | 0.669 | 0.545 | 0.527 |
| Hidden size: 100 | 0.567 | 0.447 | 0.488 | 0.904 | 0.580 | 0.608 |
| Hidden size: 256 | 0.587 | 0.527 | 0.546 | 0.983 | 0.931 | 0.956 |

Table 4: Model comparison on ROUGE-L and TAG Dependency Overlap scores.

## 2.4 Results

The reference encoder, supertag encoder, and paraphrase decoder all have the same number of hidden units. We trained the full model with three different hidden sizes: 50, 100, and 256.

Table 3 gives a comparison of model performance on word-for-word, supertag-for-supertag, and word-for-supertag accuracy. Across all models, word-for-supertag accuracy was slightly higher than simple word-for-word accuracy. This indicates that getting the correct supertag was important and helpful when predicting the correct model output. The model with the largest hidden size, 256, performed the best on all three metrics. The particularly high supertag accuracy of this model indicates that it learned to follow the syntactic constraint dictated by the desired supertag sequence given with the input.

Table 4 gives a comparison of model performance using ROUGE-L and the TAG dependency overlap scores described in Section 2.3. A lower ROUGE-L score indicates a less similar sentence in terms of word choice and word order, so this is preferred in a good paraphrase. A higher TAG dependency overlap score indicates that the paraphrase successfully reproduced the main verb (Root) and the correct deep syntactic roles of subject and object relative to the main verb.

In terms of ROUGE-L scores, the model with the smallest hidden size had the lowest precision, recall, and f-score, indicating that these outputs were the most different from the inputs in terms of word choice and word order. However,

| Reference Sentence | Desired Supertags | Model Output |
|---|---|---|
| the curious penguin kicked a woman . | t1 t3 t23 t331 tCO t1 t36 t3 t26 | a woman was kicked by the curious penguin . |
| the shy scientist was loved by the goofy moose . | t1 t2 t3 t27 t1 t36 t3 t26 | the goofy elk loved the shy scientist . |
| by the barn , a cat was hated by the big sheep . | t1 t3 t23 t331 tCO t1 t36 t3 t0 t1 t3 t26 | a guy was hated by the big sheep by the barn . |
| the silly crocodile near the mall must love the shy model . | t1 t36 t3 t4 t1 t3 t749 t31 t68 t1 t36 t3 t26 | the silly crocodile near the room the model must the shy model . |

Table 5: Sample model outputs for reference sentences and the associated supertag sequence. Model errors are highlighted in red.

the poor performance of this model on the TAG dependency overlap metrics indicates that while the outputs were more syntactically diverse, they were not faithful representations of the semantic content of the input sentence. The model with hidden size 100 had slightly higher ROUGE-L scores indicating outputs more syntactically similar to the inputs, but performed better on maintaining the key TAG dependencies. In particular, the medium-sized model was able to produce the correct root verb 90.4% of the time, but still produced the correct subject and object 58.0% and 60.8% of the time respectively on the full evaluation test set.

The model with the largest size performed the best in terms of TAG dependencies, successfully producing the correct root in 98.3% of sentences, the correct subject in 93.1%, and the correct object in 95.6%. This model was close to perfect in producing sentences with the same root, subject, and object as the input sentence, and it also had similar ROUGE-L to the gold paraphrases. The high word-for-word and supertag-for-supertag accuracy of this model from Table 3 suggest that this model learned to accurately produce the gold paraphrases exactly. Some sample successful and unsuccessful model outputs are shown in Table 5. Most of the errors in the model output come from selecting the wrong noun, repeating nouns and noun phrases, and dropping key words from the input. The overall success of this model demonstrates the effectiveness of the SCPN architecture and the use of elementary trees as syntactic constraints on the output.

## 2.5 Extension: Supertag Translation

An important part of this model architecture is the syntactic constraint encoded in the supertag sequence for the desired paraphrase. However, to reliably produce good paraphrases for novel inputs, the problem remains of what supertag sequence to provide as a syntactic guide for the output. One approach to this problem is to treat the syntactic transformation involved during paraphrase generation as *translation* of the supertag sequence of the input.

| Model | ROUGE-L | | | TAG Dependency Overlap | | |
|---|---|---|---|---|---|---|
| | Precision | Recall | F-Score | Root | Subject | Object |
| Hidden size: 256 + Gold | 0.587 | 0.527 | 0.546 | 0.983 | 0.931 | 0.956 |
| Hidden size: 256 + Translation | 0.647 | 0.546 | 0.582 | 0.965 | 0.927 | 0.929 |

Table 6: Model comparison for ROUGE-L and TAG dependency overlap for the model with hidden size 256 using gold paraphrase supertags and the model with hidden size 256 using translated supertags.

Using the same reference-paraphrase corpus described in Section 2.2, we treat the supertag sequences of the reference sentences as the source language and the supertag sequences of the paraphrase sentences as the target and train a simple sequence-to-sequence machine translation model using OpenNMT (Klein et al., 2017). For evaluation, we first translate the supertag sequences of the evaluation reference sentences using the OpenNMT model, and then use these as the syntactic constraints for paraphrase generation using the model discussed in section 2.1 with hidden size 256. We then evaluated the model outputs using the translated supertag sequences with the same ROUGE-L and TAG dependency overlap metrics as above. Since the paraphrases are generated from translated sequences of supertags, there are no 'gold' paraphrases to compare to, so we do not compute word-for-word, supertag-for-supertag, or word-for-supertag accuracy for the model outputs.

Table 6 reports ROUGE-L and TAG dependency overlap for the original model hidden size 256 provided gold paraphrase supertag sequences and provided with translated paraphrase supertag sequences. The ROUGE-L values are higher for the model using translated supertags, indicating that these outputs were more similar to the reference sentences. The model using translated supertags performed slightly below the model using gold supertags on the TAG dependency overlap metrics. Table 7 gives examples of some successful and unsuccessful outputs of the model using translated supertags. In general, the generated paraphrases are often shorter than the reference sentences. Prepositional phrases are often omitted, but these are not essential for an acceptable paraphrase. Other errors include omitting key words and repeating nouns and noun phrases, similar to the errors of the model using gold paraphrases above.

Overall, while the model using translated supertags produces slightly lower-quality paraphrases that were less syntactically diverse and semantically faithful to the input, the model can still perform reasonably well at the paraphrase gen-

| Reference Sentence | Model Output |
|---|---|
| at the mall , the happy bear stalked a shy man . | the happy bear stalked a shy man . |
| a tiny woman must stalk a tiny penguin by a house . | a tiny woman has got to stalk a tiny penguin . |
| the model was loved by the small bird . | the little bird was the model . |
| the duck hated a goofy cow by the room . | a cow cow by the room was hated by the duck . |

Table 7: Sample outputs for the model using translated supertag sequences. Model errors are highlighted in red.

eration task without the need for a set of gold paraphrases. By incorporating the syntactic transformation into the paraphrase generation process, rather than requiring a set of gold paraphrases to dictate the syntactic constraints, this model demonstrates an end-to-end method of syntactically-controlled paraphrase generation. The end-to-end system consists of a supertagger, a supertag translator, and a paraphrase generator, with only input being a reference sentence to paraphrase. The modular nature of the architecture, with the supertagger and supertag translation models trained separately from the paraphrase generation model, suggests that improvements could be made to each individual component to improve the quality of paraphrase generation. By training the supertag translator on more diverse sets of supertag sequence pairs, we could open the model up to producing even more syntactically diverse paraphrases.

# 3 Sentence Planning

## 3.1 Modified Model Architecture

In this section, we modify the model described above to simulate different paradigms of human sentence planning. The key manipulation is in the supertag encoder, which was a bidirectional LSTM in the original architecture described in Section 2.1. A bidirectional encoding of the supertag sequence is implausible as a model of sentence planning, as a human may not fully generate the planned structure of a sentence before beginning to speak. Instead of a bidirectional LSTM for supertag encoding, we use a unidirectional GRU with the same architecture as the reference sentence encoder (see Figure 5 for equations). The decoder then receives at each step the previous word $w_{t-1}$, the attention-weighted average of encoder hidden states $a_t$, and the output of the unidirectional elementary tree encoder $u_t$, and computes

$$h_t = \text{GRU}[w_{t-1}; a_t; u_t]. \tag{12}$$

By encoding the supertag sequence only from left to right, this simulates how a human might plan the structure of a sentence sequentially, perhaps beginning to utter words before the full structure of the sentence has been determined.

## 3.2 Data Manipulation

The order of elementary trees is manipulated to simulate linear sentence planning (choosing lexical items in left-to-right sentential order) or more hierarchically-driven sentence planning (choosing the verb and grammatical arguments first) that more closely matches the sentence production models discussed in Section 1.4. The linear sentence planning paradigm will use the original sequences of supertags and words for desired paraphrases as described in Section 2.2. To simulate hierarchical planning, we reorder the supertag sequence and the associated sentence for each desired paraphrase according to a top-down, left-to-right traversal of a TAG derivation tree for the sentence. This follows the linguistic intuition that when planning a sentence or phrase, we will select the head first, and then fill in the dependents.

First, the paraphrase sentences and supertags are passed to the MICA TAG Parser (Bangalore et al., 2009). The MICA parser computes a dependency parse of each sentence using the syntactic information encoded in the elementary trees, and outputs a list of TAG dependency relations connecting the words in the sentence, which can be interpreted into a derivation tree for the sentence.

The basic algorithm to reorder the paraphrase sentence $s_2$ and elementary tree sequence $t_2$ to the reordered $s_2'$ and $t_2'$ follows a generally top-down, left-to-right traversal of the derivation tree of the sentence. First, identify the main predicate, marked with the deep syntactic role of ROOT in the MICA parse, and put this word and the corresponding supertag first. Then recursively copy over the root item's dependents, starting with those on the left and their dependents. Figure 7 gives an example of the derivation tree and reordered outputs for the sentence *the cat ate a fish*.

| Linear-order sentence |
|---|
| $s_2$: the cat ate a fish |
| $t_2$: `t1 t3 t27 t1 t3` |

| Hierarchical-order sentence |
|---|
| $s'_2$: ate cat the fish a |
| $t'_2$: `t27 t3 t1 t3 t1` |

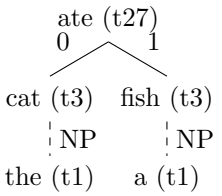


Figure 7: Example of hierarchical reordering (left) for the sentence *the cat ate a fish*, following a top-down, left-to-right traversal of the derivation tree (right).

| Model | WFW | SFS | WFS |
|---|---|---|---|
| Hidden size: 50 | | | |
| Linear | 0.708 | 0.962 | 0.715 |
| Hierarchical | 0.786 | 0.960 | 0.799 |
| Hidden size: 100 | | | |
| Linear | 0.740 | 0.922 | 0.773 |
| Hierarchical | 0.874 | 0.962 | 0.887 |

Table 8: Model comparison for different hidden sizes and training data ordering on word-for-word (WFW), supertag-for-supertag (SFS), and word-for-supertag (WFS) accuracy.

### 3.3 Results

For each hidden size (50 or 100)[2], we train the model described in Section 3.1 using either linearly-ordered or hierarchically-ordered supertags for the syntactic constraint. Because the model trained on hierarchically-ordered supertags outputs sentences in the hierarchical order shown in Figure 7, we first reverse the procedure used to generate the hierarchical order to reorder these outputs back to linear left-to-right order. This allows the supertagger and parser to process the outputs of the hierarchical model correctly.

We evaluate each model on the same metrics described in Section 2.3, comparing basic model performance on word-for-word, supertag-for-supertag, and word-for-supertag accuracy as well as ROUGE and TAG dependency overlap.

Table 8 shows the performance of each model on the word-for-word (WFW), supertag-for-supertag (SFS), and word-for-supertag (WFS) accuracy metrics. The models both had similarly high supertag-for-supertag accuracy, indicating that the unididrectional supertag encoding was generally successful at enforcing the syntactic constraint on the output. The models trained to produce paraphrases in hierarchical order was more successful at producing the desired paraphrase word-for-word. This implies some advantage to providing syntactic information in a hierarchical way over linear order.

[2]We also ran each model with hidden size 256, but these models had evaluation errors.

| Model | ROUGE-L Precision | ROUGE-L Recall | ROUGE-L F-Score | TAG Dependency Overlap Root | TAG Dependency Overlap Subject | TAG Dependency Overlap Object |
|---|---|---|---|---|---|---|
| Gold Paraphrases | 0.594 | 0.534 | 0.693 | 0.981 | 0.936 | 0.963 |
| Hidden size: 50 | | | | | | |
| Linear | 0.520 | 0.391 | 0.435 | 0.727 | 0.362 | 0.359 |
| Hierarchical | 0.587 | 0.447 | 0.496 | 0.889 | 0.793 | 0.756 |
| Hidden size: 100 | | | | | | |
| Linear | 0.428 | 0.391 | 0.406 | 0.876 | 0.431 | 0.431 |
| Hierarchical | 0.666 | 0.611 | 0.633 | 0.938 | 0.884 | 0.873 |

Table 9: Model comparison on ROUGE-L and TAG Dependency Overlap scores.

Table 9 gives a comparison of model performance using ROUGE-L and the TAG dependency overlap scores described in Section 2.3. A lower ROUGE-L score indicates a less similar sentence in terms of word choice and word order, so this is preferred in a good paraphrase. A higher TAG dependency overlap score indicates that the paraphrase successfully reproduced the main verb (Root) and the correct deep syntactic roles of subject and object relative to the main verb.

In terms of ROUGE-L scores, the models trained on linear-order data had lower scores than models of the same size trained on hierarchical-order data. This indicates that the outputs of the linear-order models were less similar to the input, preferable in good paraphrases. However, the TAG dependency overlap scores for these models indicate that the models trained on linearly-ordered data did a poor job of maintaining the key semantic information from the input. The proportion of sentences for these models that produced the correct subject and object were particularly low. This fits with the prediction that the unidirectional encoding of supertags will result in the model producing the subject before receiving information about the verb that might affect the order of noun arguments in the sentence.

Table 10 compares model performance in terms of TAG dependency overlap on two subsets of the data, one including only active-passive and passive-active pairs, and the other including only active-active and passive-passive pairs. While the majority (72%) of pairs involving actives and passives are active-passive or passive-active transformations, the other (28%) in the subset do not include the transformation, so the model cannot be completely successful by simply learning to perform the transformation every time.

| Model | Active-Passive TAG Dependency Overlap | | | Active-Active TAG Dependency Overlap | | |
|---|---|---|---|---|---|---|
| | Root | Subject | Object | Root | Subject | Object |
| Gold Paraphrases | 0.999 | 0.967 | 0.991 | 0.999 | 0.944 | 0.968 |
| Hidden size: 50 | | | | | | |
| Linear | 0.936 | 0.368 | 0.374 | 0.964 | 0.351 | 0.375 |
| Hierarchical | 0.995 | 0.824 | 0.826 | 0.992 | 0.810 | 0.686 |
| Hidden size: 100 | | | | | | |
| Linear | 0.955 | 0.448 | 0.459 | 0.842 | 0.437 | 0.432 |
| Hierarchical | 0.993 | 0.951 | 0.964 | 0.990 | 0.881 | 0.890 |

Table 10: Model comparison for TAG dependency overlap on active-passive and passive-active sentences (left) and active-active and passive-passive sentences (right).

Within each model, performance on the active-passive subset was generally slightly above performance on the active-active subset. This is likely due to the significantly greater frequency of active-passive pairs compared to active-active pairs. Since the difference in performance is relatively small, this indicates that the models were most likely not simply changing the voice for every active or passive sentence, and rather followed the supertag sequence for each sentence to correctly predict whether or not the transformation should occur. As with the full evaluation set, the models trained on hierarchically-ordered data outperformed those trained on linearly-ordered data.

The models trained on hierarchically-ordered data performed better at producing syntactically diverse while semantically faithful paraphrases compared to the models trained on linearly-ordered data. This supports the prediction that providing the syntactic constraint in a more efficient way relative to the hierarchical linguistic structure of the desired output sentence will result in better paraphrases.

# 4 Discussion

In Section 2, we outlined the performance of several models trained on our limited paraphrase generation task. While performance differed depending on the model architecture, the models for the most part were able to produce reasonable paraphrases of English sentences. The model with the largest hidden

size performed best, generating paraphrases that were high in semantic fidelity to the reference sentences and comparable in syntactic diversity to the gold paraphrases. The performance of this model indicates that TAG elementary trees can be successfully used as syntactic constraints for paraphrase generation. In addition, Section 2.5 demonstrates a method for end-to-end paraphrase generation by incorporating supertag translation as an intermediate step. This allows the model to generalize to new domains and inputs for which the desired sequence of elementary trees for the paraphrase is not explicitly known.

In Section 3, we demonstrated how a similar model architecture could be used to explore two paradigms for syntactic planning in sentence production. Using a unidirectional supertag encoder, we were able to simulate sequential syntactic planning, and provided the models with two different methods for ordering the syntactic constraint. Organizing the desired paraphrase supertags hierarchically did boost performance over organizing them linearly. This supports the hypothesis that due to the linear, left-to-right nature of the unidirectional supertag encoding, in order to produce the correct arguments at the right time, the model must learn the most important and syntactically relevant information first. This supports the sentence production models discussed in Section 1.4 in that it seems easier to produce a desired sentence when syntactically relevant information such as the main verb is selected first. While more research into human sentence planning is necessary, this work provides some computational evidence supporting hierarchical organization of syntactic planning during sentence production.

# 5 Future Work

A significant limitation of this work is the artificial nature of the data used for training and testing. While the simplified data did ensure a controlled environment with quality pairs of paraphrases, the training corpus used hardly represents the vast diversity of possible paraphrase pairs in English. In particular, the dataset was significantly biased towards active-passive and passive-active transformations. The conclusions made from the results of the model comparison must be considered in the context of the simplified task environment. In future work, we plan to use the same model architecture trained on real-world data, such as question pairs from Quora (Iyer et al., 2017) or image captions from Flickr (Plummer et al., 2015).

It would also be interesting to explore how the unidirectional elementary tree encoding with the two ordering paradigms would perform in a verb-final language such as German or Japanese. We might expect the linear order to result in very low model performance, while the hierarchical order should result in similarly high performance as in English.

## 6 Acknowledgements

This work would not have been possible without the endless support and encouragement of Professor Bob Frank in the Linguistics Department at Yale. I would also like to thank the other members of Computational Linguistics at Yale as well as the Yale Cognitive Science Department for supporting me through this process. Finally, thanks to my family and friends for putting up with never-ending conversations about paraphrases and buggy Python code.

# References


Bahdanau, D., Cho, K., and Bengio, Y. (2015). Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Bangalore, S., Boullier, P., Nasr, A., Rambow, O., and Sagot, B. (2009). Mica: a probabilistic dependency parser based on tree insertion grammars.

Bangalore, S. and Joshi, A. K. (1999). Supertagging: An approach to almost parsing. *Comput. Linguist.*, 25(2):237–265.

Bock, K. and Levelt, W. J. M. (1994). Language production : Grammatical encoding.

Chen, M., Tang, Q., Wiseman, S., and Gimpel, K. (2019). 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.

Cho, K., van Merrienboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation.

Fellbaum, C. (1998). *WordNet: An Electronic Lexical Database*. Bradford Books.

Gan, W. C. and Ng, H. T. (2019). Improving the robustness of question answering systems to question paraphrasing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6065–6075, Florence, Italy. Association for Computational Linguistics.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9:1735–80.

Hu, J. E., Rudinger, R., Post, M., and Durme, B. V. (2019). Parabank: Monolingual bitext generation and sentential paraphrasing via lexically-constrained neural machine translation. *Proceedings of the AAAI Conference on Artificial Intelligence*, 33:6521–6528.

Iyer, S., Dandekar, N., and Csernai, K. (2017). First quora dataset release: Question pairs. *data. quora. com.*

Iyyer, M., Wieting, J., Gimpel, K., and Zettlemoyer, L. (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.

Joshi, A. K., Levy, L. S., and Takahashi, M. (1975). Tree adjunct grammars. *Journal of computer and system sciences*, 10(1):136–163.

Kajiwara, T. (2019). Negative lexically constrained decoding for paraphrase generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6047–6052, Florence, Italy. Association for Computational Linguistics.

Kasai, J., Frank, R., McCoy, R. T., Rambow, O., and Nasr, A. (2017). Tag parsing with neural networks and vector representations of supertags. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Kempen, G. and Hoenkamp, E. (1987). An incremental procedural grammar for sentence formulation. *Cognitive Science*, 11(2):201–258.

Klein, G., Kim, Y., Deng, Y., Senellart, J., and Rush, A. M. (2017). Opennmt: Open-source toolkit for neural machine translation. *arXiv preprint arXiv:1701.02810*.

Levelt, W. J. M. (1989). *Speaking: From intention to articulation*. MIT Press, Cambridge, MA.

Li, Z., Jiang, X., Shang, L., and Liu, Q. (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.

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

Lin, C.-Y. and Och, F. J. (2004). Automatic evaluation of machine translation quality using longest common subsequence and skip-bigram statistics. In *Proceedings of the 42nd Annual Meeting of the Association for Computational Linguistics (ACL-04)*, pages 605–612, Barcelona, Spain.

Loper, E. and Bird, S. (2002). NLTK: the natural language toolkit. *CoRR*, cs.CL/0205028.

Manning, C., Surdeanu, M., Bauer, J., Finkel, J., Bethard, S., and McClosky, D. (2014). The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland. Association for Computational Linguistics.

Mao, H.-R. and Lee, H.-Y. (2019). Polly want a cracker: Analyzing performance of parroting on paraphrase generation datasets. 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 5960–5968, Hong Kong, China. Association for Computational Linguistics.

Momma, S., Slevc, L. R., and Phillips, C. (2016). The timing of verb selection in japanese sentence production. *Journal of Experimental Psychology: Learning, Memory, and Cognition*, 42(5):813–824.

Plummer, B. A., Wang, L., Cervantes, C. M., Caicedo, J. C., Hockenmaier, J., and Lazebnik, S. (2015). Flickr30k entities: Collecting region-to-phrase correspondences for richer image-to-sentence models. In *The IEEE International Conference on Computer Vision (ICCV)*.

Qian, L., Qiu, L., Zhang, W., Jiang, X., and Yu, Y. (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 3173–3182, Hong Kong, China. Association for Computational Linguistics.

Roy, A. and Grangier, D. (2019). Unsupervised paraphrasing without translation. *CoRR*, abs/1905.12752.

Schriefers, H., Teruel, E., and Meinshausen, R. (1998). Producing simple sentences: Results from picture–word interference experiments. *Journal of Memory and Language*, 39(4):609 – 632.

Wieting, J. and Gimpel, K. (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.

Yang, Q., Huo, Z., Shen, D., Cheng, Y., Wang, W., Wang, G., and Carin, L. (2019). An end-to-end generative architecture 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 3132–3142, Hong Kong, China. Association for Computational Linguistics.

Zhou, Z., Sperber, M., and Waibel, A. (2019). Paraphrases as foreign languages in multilingual neural machine translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 113–122, Florence, Italy. Association for Computational Linguistics.