GRS: Combining Generation and Revision in Unsupervised Sentence Simplification

Mohammad Dehghan[1], **Dhruv Kumar**[2], **Lukasz Golab**[1]
[1]University of Waterloo
[2]Grammarly
{m25dehgh,lgolab}@uwaterloo.ca
dhruv.kumar@grammarly.com

Abstract

We propose GRS: an unsupervised approach to sentence simplification that combines text generation and text revision. We start with an iterative framework in which an input sentence is revised using explicit edit operations, and add paraphrasing as a new edit operation. This allows us to combine the advantages of generative and revision-based approaches: paraphrasing captures complex edit operations, and the use of explicit edit operations in an iterative manner provides controllability and interpretability. We demonstrate these advantages of GRS compared to existing methods on the Newsela and ASSET datasets.

1 Introduction

Text simplification is the task of reducing the complexity and improving the readability of text while preserving its meaning. This is beneficial for persons with reading disabilities (Evans et al., 2014), non-native speakers, people with low literacy, and children. Furthermore, other NLP tasks can use simplification as a pre-processing step, such as summarization (Klebanov et al., 2004), parsing (Chandrasekar et al., 1996), and machine translation (Štajner and Popovic, 2016).

Sentence simplification models can be categorized into *generative* and *revision-based*. Generative approaches produce a simple sentence from a complex sentence in one step, in an auto-regressive way (Zhang and Lapata, 2017; Guo et al., 2018; Kriz et al., 2019; Surya et al., 2019; Martin et al., 2020a). Revision-based methods iteratively edit a given sentence using a sequence of edit operations such as word deletion (Alva-Manchego et al., 2017; Dong et al., 2019; Kumar et al., 2020; Agrawal et al., 2021). While generative models learn complex edit operations implicitly from data, the explicit edit operations performed by revision-based approaches can provide more control and interpretability.

Simplification methods can also be categorized as supervised or unsupervised. Supervised methods tend to have better performance, but require aligned complex-simple sentence pairs for training (Zhang and Lapata, 2017; Guo et al., 2018; Kriz et al., 2019; Martin et al., 2020a,b; Maddela et al., 2021). Unsupervised methods do not need such training data but do not perform as well (Surya et al., 2019; Kumar et al., 2020; Zhao et al., 2020).

We propose GRS: a new approach to bridge the gap between generative and revision-based methods for unsupervised sentence simplification. The insight is to introduce *paraphrasing* as an edit operation within an iterative revision-based framework. For paraphrasing, we use a fine-tuned BART model (Lewis et al., 2020) with lexically-constrained decoding (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019a). This decoding technique allows us to select words from the initial sentence that must be changed in the paraphrased sentence (otherwise, paraphrasing an entire sentence reduces to a pure generative model). To avoid the computational overhead of repeatedly performing constraint-based decoding using various combinations of words to paraphrase, GRS includes a complex component detector to identify the most appropriate words to paraphrase. The code is available at https://github.com/imohammad12/GRS.

GRS is unsupervised in the sense that it does not require aligned complex-simple sentence pairs, but it uses supervised models. The paraphrasing model requires paraphrasing corpora, and the complex component detector requires two unlabeled corpora, one containing more complex sentences than the other. However, collecting paraphrasing data and unaligned simplification data is simpler than collecting aligned complex-simple pairs.

2 Related Work

Early work on simplification relied on rules, e.g., to split or shorten long sentences (Chandrasekar and

Findings of the Association for Computational Linguistics: ACL 2022, pages 949 - 960

Srinivas, 1997; Carroll et al., 1998; Vickrey and Koller, 2008). Later work treated simplification as a monolingual phrase-based machine translation (MT) task (Coster and Kauchak, 2011; Wubben et al., 2012), with syntactic information added, such as constituency trees (Zhu et al., 2010). Recent work, reviewed below, leverages neural models in a *generative* and *revision-based* manner.

Supervised Generative Methods employ Seq2Seq models to learn simplification operations from aligned complex-simple sentence pairs (Nisioi et al., 2017). Building on a Seq2Seq model, Zhang and Lapata (2017) used reinforcement learning to optimize a reward based on simplicity, fluency and relevance. Recent methods build on transformer (Vaswani et al., 2017) models, by integrating external databases containing simplification rules (Zhao et al., 2018), using an additional loss function to generate diverse outputs (Kriz et al., 2019), combining syntactic rules (Maddela et al., 2021), and conditioning on length and syntactic and lexical complexity features (Martin et al., 2020a).

Unsupervised Generative Methods rely on non-aligned complex and simple corpora. Zhao et al. (2020) adopted a back-translation framework, whereas Surya et al. (2019) used an unsupervised style transfer paradigm. Martin et al. (2020b) used a pre-trained BART model fine-tuned on paraphrased sentence pairs.

Controllable Generative Methods produce outputs at specified grade levels (Scarton and Specia, 2018; Nishihara et al., 2019), or apply syntactic or lexical constraints on the generated sentences (Martin et al., 2020a,b). However, these models do not provide any insights into the simplification process.

Supervised Revision-Based Methods use complex-simple sentence pairs to learn where to apply edit operations. Alva-Manchego et al. (2017) use keep, replace, and delete operations. Some recent work used iterative non-autoregressive models to edit sentences by either predicting token-level edit-operations (Omelianchuk et al., 2021) or using a fixed pipeline of edit operations (Agrawal et al., 2021). Dong et al. (2019) proposed a hybrid method with explicit edit operations in an end-to-end generative model.

Unsupervised Revision-Based Methods such as Narayan and Gardent (2016) apply a pipeline of edit operations in a fixed order. Kumar et al. (2020) presented an unsupervised revision-based approach by modelling text simplification as an unsupervised search problem. While GRS also uses a revision-based framework and an unsupervised search strategy, we integrate a generative paraphrasing model into the framework to leverage the strengths of both text generation and text revision approaches.

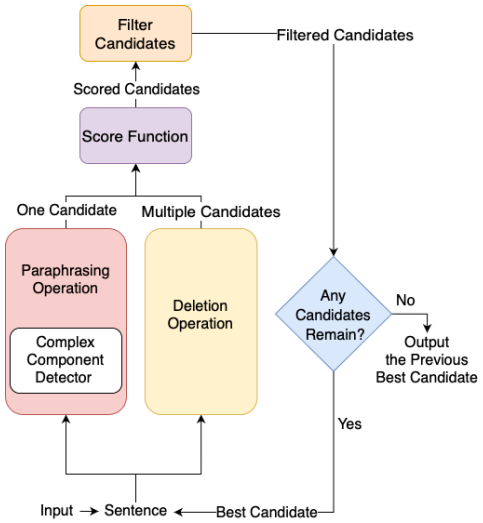

Figure 1: An overview of GRS. Given a complex input sentence, simplifications are iteratively produced via paraphrasing and deletion, with paraphrasing guided by the complex component detector. Sentences passing a filter (Equation 4) are candidates for input to the next iteration.

3 GRS Model

3.1 Overview

Our solution, GRS, iteratively revises a given complex sentence by applying edit operations on sentence fragments. In each iteration, multiple candidate simplifications are produced and evaluated using a scoring function (Section 3.5), and the best candidate is selected (Section 3.6). The selected sentence acts as the input to the next iteration. This process continues until none of the candidate sentences are simpler than the input sentence.

GRS uses two edit operations: *paraphrasing* (Section 3.2; guided by the complex component detector described in Section 3.3) and *deletion* (Section 3.4). The scoring function (Section 3.5) guides our search for best simplifcation, using soft and hard constraints on simplicity, linguistic acceptability, and meaning preservation.

In Section 3.6, we explain how paraphrasing and deletion work in an iterative search framework, how candidate sentences are selected, and when the algorithm terminates. Figure 1 gives an overview of GRS, which is explained further in Section 3.6.

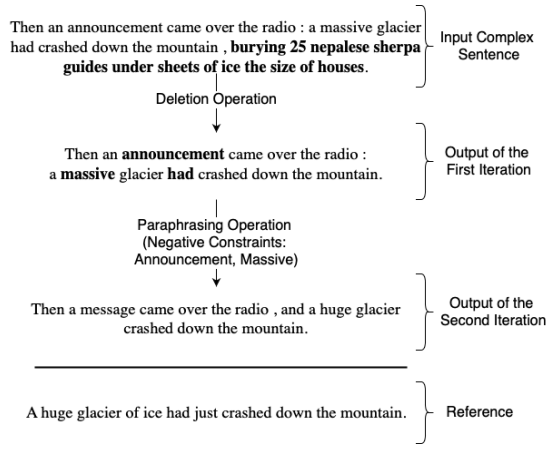

Figure 2: Two iterations of edit operations: deletion, then paraphrasing to simplify the complex fragments ("announcement" and "massive") identified by the complex component detector and given as negative constraints to the paraphrasing model. This example demonstrates the interpretability of GRS through building a simplification path leading to the final sentence.

3.2 Paraphrasing Operation

We use a pre-trained BART model (Lewis et al., 2020), fine-tuned on a small subset of ParaBank 2 paraphrasing dataset (Hu et al., 2019b); however, any paraphrasing auto-regressive model can be used instead. During inference, we use lexical-constrained decoding (Hokamp and Liu, 2017; Post and Vilar, 2018; Hu et al., 2019a) to place negative constraints on complex words and phrases in the input sentence. Negative constraints are words that the paraphrasing model is forced not to generate during decoding. Figure 2 shows an example in which an input sentence was paraphrased to exclude two complex words (negative constraints): "massive" and "announcement". We explain how to choose negative constraints below, with the help of the complex component detector.

3.3 Complex Component Detector

Constrained decoding is computationally more expensive than greedy and beam search decoding. In GRS, before paraphrasing a sentence, the complex component detector predicts the best negative constraints; then the sentence and the predicted negative constraints are given to the paraphrasing model to generate a new candidate sentence. As a result, the paraphrasing operation is called only once per iteration of GRS, using the predicted negative constraints, avoiding the expensive process of repeatedly paraphrasing the input using different combinations of negative constraints.

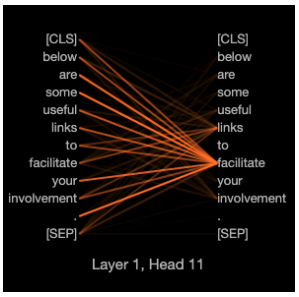

Figure 3: One of the attention matrices of the DeBERTa complex-simple classifier (head 11 of the second layer). Attention weights are reflected by color intensity. The input sentence in this example is "below are some useful links to facilitate your involvement." We used BertViz (Vig, 2019) to visualize attention weights.

We implemented the complex component detector as a complex-simple classifier that gives a simplicity probability to a given sentence. We only require two corpora with different complexity levels to train this classifier. Since aligned complex-simple sentence pairs are not required, this classifier can be trained on any pair of corpora with different complexity levels. Reid and Zhong (2021) showed that it is possible to extract style-specific sections of a sentence using the attention layers of a style classifier. Similarly, we use the attention layers of our complex-simple classifier to extract the complex components from a given input sentence.

We fine-tune the pre-trained DeBERTa model (He et al., 2020) as our complex-simple classifier. Figure 3 illustrates one of the attention heads of the second layer of DeBERTa. This visualization shows that the word "faciliate" was attended to more than the other words in the given sentence. We use this intuition and devise a formula (Equations 1 and 2 below) to detect complex words by analyzing attention weights.

BERT (Devlin et al., 2019) and its extensions (e.g. DeBERTa) add a [CLS] token to the beginning and a [SEP] token to the end of each sentence (as shown in Figure 3). In these models, the hidden states of the [CLS] token in the last layer are used for classification tasks. In our complex-simple classifier, we found that the attention paid by the [CLS] token in the second layer to other words in the sentence can help us detect complex components. Equations 1 and 2 demonstrate how we extract complex components from attention head matrices of the second layer of the classifier. Here, $\mathbf{A}_{h,i}^{[CLS]}$ refers to the amount of attention the [CLS] token in the hth attention head of the second layer

pays to the ith token of the input sentence. N and H refer to the length of the input sentence and the number of attention heads, respectively. c_i defines whether the ith token is complex or not. If $c_i = 1$, then this token will be set as a negative constraint. $\bar{T}$ is a threshold used for finding complex tokens. In the example demonstrated in Figure 3, only c_8, which refers to the word "facilitate", is a complex token.

$$\bar{T} = \frac{\sum_{h=0}^{H-1} \sum_{i=0}^{N-1} \mathbf{A}_{h,i}^{[CLS]}}{N} \quad (1)$$

$$c_i = \begin{cases} 1, & \text{if } \sum_{h=0}^{H-1} \mathbf{A}_{h,i}^{[CLS]} \geq \bar{T} \\ 0, & \text{otherwise} \end{cases} \quad (2)$$

3.4 Deletion Operation

Deletion aims to remove peripheral information to make sentences simpler, and is composed of two sub-operations: removal and extraction. Inspired by Kumar et al. (2020), we use the *constituency tree* of the input sentence to obtain all constituents from different depths of the parse tree. These constituents can be deleted (removal) or selected as a simplified candidate sentence (extraction). The removal sub-operation creates new candidate sentences by removing each of these phrases from the input sentence. The extraction sub-operation selects phrases as candidate sentences, which helps the model extract the main clause and drop peripheral information. The example in Figure 2 drops the phrase *"burying 25 nepalese sherpa guides under sheets of ice the size of houses"* from the complex sentence since it is not the main clause.

3.5 Scoring Function

Candidates generated by our two edit operations may not be correct in terms of linguistic acceptability. Furthermore, important information from the original sentence may have been removed. We use a score function to filter out non-grammatical candidates or sentences that are not conceptually similar to the original sentence. The score function is composed of three important components.

Meaning Preservation (H_{mp}): First, we use the method proposed in Reimers and Gurevych (2019) to obtain the semantic representations of the sentences. We then use the cosine similarity measure between the representations of the original and the generated candidate sentence. Our meaning preservation measure acts as a hard filter. A hard filter assigns a zero score to candidate sentences that do not pass a certain threshold.

Linguistic Acceptability (H_{la}): By removing some components of a complex input sentence, the output sentence may become nonsensical. To check the linguistic acceptability of the generated sentences, we train a classifier on the CoLA (the corpus of linguistic acceptability) (Warstadt et al., 2019) dataset. This classifier measures the probability that a given sentence is grammatical. This module, like the meaning preservation module, is used as a hard filter in the score function.

Simplicity (S_{simp}): This is a soft constraint, for which we use the complex-simple classifier mentioned in Section 3.3, which computes the simplicity probability of a sentence.

These three measures together evaluate the quality of each candidate sentence, as shown in Equation 3. In this equation, S, S_{simp}, H_{la}, H_{mp}, c, and o refer to the score function, simplicity module, linguistic acceptability hard filter, meaning preservation hard filter, candidate sentence, and the original sentence, respectively.

$$S(c) = S_{simp}(c) * H_{mp}(c, o) * H_{la}(c) \quad (3)$$

3.6 Simplification Search

Our unsupervised search method is inspired by Kumar et al. (2020), but with different simplification operations and a different score function. Given a complex input sentence, paraphrasing and deletion operations generate candidate sentences separately. In each iteration, the paraphrasing operation creates only one candidate sentence, as described in Section 3.2, whereas the deletion operation generates multiple candidate sentences (Section 3.4). Candidates sentences are then evaluated according to the scoring function. Given a score for each candidate sentence, we filter out those candidates that do not improve the score of the input sentence by some threshold. The threshold depends on the edit operation that the candidate sentence has been created from. In equation 4, t_{op} is the threshold associated with operator op. S, c, and c' refer to the score function, the candidate sentence, and the input sentence in the current iteration, respectively.

$$S(c) > S(c') * t_{op} \quad (4)$$

Finally, at the end of each iteration, out of the

remaining sentences (that are not filtered out), we select the one with the highest score.

4 Experiments

This section discusses the experimental setup (Sections 4.1 through 4.4), a comparison of GRS with existing approaches (Section 4.5), a controllability study (Section 4.6), an evaluation of the complex component detector (Section 4.7), an analysis of the simplification search (Section 4.8), and a human evaluation (Section 4.9).

4.1 Data

We use the Newsela (Xu et al., 2015) and ASSET datasets (Alva-Manchego et al., 2020) to evaluate GRS against existing simplification methods. Newsela contains 1840 news articles for children at five reading levels. We use the split from Zhang and Lapata (2017), containing 1129 validation and 1077 test sentence pairs. ASSET includes 2000 validation and 359 test sentences pairs. Each sentence has ten human-written references.

4.2 Training Details

Paraphrasing Model: We fine-tune a pre-trained BART model (Lewis et al., 2020) implemented by Wolf et al. (2020). To do this, we use a subset of the ParaBank 2 paraphrasing dataset (Hu et al., 2019b) containing 47,000 pairs. We observe that a conservative paraphrasing model helps us to control the generated output. This is because such a model is better at specifically changing only words provided as negative constraints. Thus, for fine-tuning the BART model, we select paraphrasing sentence pairs that are semantically similar to each other. For calculating the semantic similarity of sentence pairs, we use the model from Reimers and Gurevych (2019) to obtain sentence embeddings, and then we use cosine-similarity to find the most similar sentence pairs.

The BART model is composed of a 12-layer encoder and a 12-layer decoder, each layer containing 16 attention heads. The model's hidden size is 1024, and the tokenizer vocabulary size is 50265. We use a batch size of 8 (per device). It took approximately 1.5 hours to fine-tune the model using three NVIDIA 2080 Ti GPUs.

Complex-Simple Classifier: We use a pre-trained DeBERTa (He et al., 2020) model implemented by Wolf et al. (2020). This model is composed of a 12-layer self-attentional encoder, each layer containing 12 attention heads. The model's hidden size is 768, and the tokenizer vocabulary size is 30522. To fine-tune the DeBERTa model for the binary classification task, we use the Newsela-Auto dataset (Jiang et al., 2020). To train the classifier, we use the AdamW (Loshchilov and Hutter, 2019) optimizer with a learning rate of 5×10^{-5} and a batch size of 16. Note that we do not use the alignment between complex-simple sentence pairs in the Newsela-Auto dataset. Thus, our complex-simple classifier can be trained on any text corpora with different complexity levels. It took approximately one hour to fine-tune the classifier using a single NVIDIA 2080 Ti GPU. The accuracy of this classifier is 78.46.

Meaning Preservation Module of the Scoring Function: To obtain contextual embeddings of sentences, we use the SentenceTransformers (Reimers and Gurevych, 2019) framework, specifically, the paraphrase-mpnet-base-v2 pre-trained model.

Linguistic Acceptability Module of the Scoring Function: To score the linguistic acceptability of a sentence, we fine-tune a pre-trained DeBERTa model (He et al., 2020) for a binary classification task on the CoLA (the corpus of linguistic acceptability) (Warstadt et al., 2019) dataset. It contains 10,657 sentences, each labelled either as grammatical or ungrammatical. The configuration and training hyperparameters of this classifier are the same as the complex-simple classifier explained above. It took approximately 30 minutes to fine-tune the model using a single NVIDIA 2080 Ti GPU. On the validation set, the accuracy of the model is 79.33.

Simplification Search and Score Function: The threshold associated with paraphrasing (t_{par}) is 0.8, and the thresholds related to the removal (t_{dl-rm}) and extraction (t_{dl-ex}) sub-operations of the deletion operation are 1.1, and 1.25, respectively. The score function's meaning preservation (H_{mp}) and linguistic acceptability (H_{la}) thresholds are 0.7 and 0.3, respectively. We obtained these values using the validation set. These values are used for both ASSET and Newsela datasets.

4.3 Evaluation Metrics

To evaluate GRS and other models, we use SARI (Xu et al., 2016) as our primary metric. SARI (System output Against References and against the Input sentence) evaluates the quality of the output text by calculating how often the output text correctly keeps, inserts, and deletes n-grams from

Model	SARI ↑	Add ↑	Delete ↑	Keep ↑	FKGL ↓	Len
Identity Baseline	12.24	0.00	0.00	36.72	8.82	23.04
Unsupervised Models						
Zhao et al. (2020)	37.20	1.51	73.53	36.54	3.80	11.78
Kumar et al. (2020)	38.36	1.01	77.58	36.51	**2.81**	9.61
Martin et al. (2020b)	38.29	**4.44**	76.02	34.42	4.65	12.49
GRS: DL (RM + EX)	37.52	0.66	69.45	**42.44**	3.93	12.64
GRS: PA	36.42	3.44	69.55	36.28	5.79	19.08
GRS: PA + DL (RM + EX)	**40.01**	3.06	**80.43**	36.53	3.20	11.72
Supervised Models						
Narayan and Gardent (2014)	34.73	0.77	73.22	30.21	4.52	12.40
Zhang and Lapata (2017)	38.03	2.43	69.47	**42.20**	4.78	14.36
Dong et al. (2019)	39.28	2.13	77.17	38.53	3.80	10.92
Zhao et al. (2020)	39.14	2.80	74.28	40.34	4.11	11.63
Martin et al. (2020b)	**41.20**	**6.02**	**81.70**	35.88	**2.35**	9.22

Table 1: Comparison of supervised and unsupervised simplification models on the Newsela test set. PA and DL refer to paraphrasing and deletion, respectively. RM and EX refer to the removal and extraction sub-operations of the deletion operation. ↑ denotes the higher the value, the better. ↓ denotes the lower the value, the better.

the complex sentence, compared to the reference text, where $1 \leq n \leq 4$. We report the overall SARI score, as well as individual SARI scores corresponding to n-grams correctly added (ADD), deleted (DELETE) and kept (KEEP); the overall SARI score is the mean of these three scores. We also report the FKGL score, which only considers the output sentence, not the source and reference sentences. It is computed based on sentence length and the number of syllables for each word in the sentence. We use the EASSE package (Alva-Manchego et al., 2019) to calculate SARI and FKGL. We do not use the BLEU (Papineni et al., 2002) metric since Sulem et al. (2018) showed that BLEU does not correlate well with simplicity.

4.4 Models Tested

We evaluate GRS with different configurations: only deletion - GRS: DL(RM+EX), only paraphrasing - GRS: PA, and both deletion and paraphrasing - GRS: PA+DL(RM+EX). We also consider the complex sentence itself as a trivial baseline, denoted by 'Identity Baseline'. The ASSET dataset contains multiple references for a sentence, so we also calculate an upper bound for a given evaluation metric, which we denote as 'Gold Reference'. To calculate the 'Gold Reference' score, each reference is selected once, and the scores are calculated against others. Finally, we average across all the reference scores to obtain the final 'Gold Reference' score.

We also compare GRS with existing approaches. From unsupervised methods, we select unsupervised generative models that use Seq2Seq models Surya et al. (2019); Zhao et al. (2020). We also compare with Martin et al. (2020b), which leverages pretrained language models and a large paraphrase pair dataset, and Kumar et al. (2020), an iterative revision-based method with several explicit edit operations (deletion, lexical substitution and reordering).

From supervised methods, we start with Narayan and Gardent (2014) and Xu et al. (2016), which use phrase-based MT models. We also consider Seq2Seq generative methods: Zhang and Lapata (2017), which uses reinforcement learning, and Zhao et al. (2020); Martin et al. (2020a,b), which use Seq2Seq transformer models. Next, we select Omelianchuk et al. (2021), a recent supervised revision-based method. Finally, we consider Dong et al. (2019), a hybrid approach using explicit edit operations in a generative framework.

4.5 Evaluation Results

Tables 1 and 2 illustrate the results on Newsela and ASSET, respectively. We report the overall SARI score, the individual scores of three operations used in SARI score, the FKGL score, and the average length of the output sentences. To evaluate previous methods, we obtained their output sentences on ASSET and Newsela from the respective project Github pages or by contacting the respective authors, followed by calculating the SARI and FKGL scores using the EASSE package (described in Section 4.3). One exception is Omelianchuk et al. (2021): since they also used the EASSE package, we copied their reported ASSET scores in Table 2, but they did not report the average sentence length.

For Newsela, using paraphrasing and deletion

Model	SARI ↑	Add ↑	Delete ↑	Keep ↑	FKGL ↓	Len
Identity Baseline	20.73	0.00	0.00	62.20	10.02	19.72
Gold Reference	44.89	10.17	58.76	65.73	6.49	16.54
Unsupervised Models						
Surya et al. (2019)	35.19	0.83	45.98	58.75	7.60	16.81
Zhao et al. (2020)	33.95	1.99	42.09	57.77	7.51	18.80
Kumar et al. (2020)	36.67	1.29	51.33	57.40	7.33	16.56
Martin et al. (2020b)	**42.42**	**7.15**	61.32	**58.77**	7.49	16.36
GRS: DL (RM + EX)	37.90	0.89	62.32	50.50	4.17	11.18
GRS: PA	40.41	7.00	62.37	51.88	6.70	17.94
GRS: PA + DL (RM + EX)	37.40	3.89	**67.46**	40.85	**3.45**	10.69
Supervised Models						
Narayan and Gardent (2014)	34.65	1.3	59.24	43.41	**5.18**	10.95
Xu et al. (2016)	37.11	5.07	45.21	61.06	7.95	20.50
Zhang and Lapata (2017)	36.59	2.38	50.10	57.30	7.66	14.37
Zhao et al. (2018)	38.67	4.36	51.37	60.29	7.73	18.36
Dong et al. (2019)	34.95	2.40	42.69	59.73	8.38	16.49
Martin et al. (2020a)	40.13	6.53	50.84	**62.99**	7.29	19.49
Zhao et al. (2020)	35.15	2.22	45.32	57.91	7.83	16.14
Martin et al. (2020b)	**44.05**	**10.93**	61.91	59.30	6.13	18.49
Omelianchuk et al. (2021)	43.21	8.04	**64.25**	57.35	6.87	——-

Table 2: Comparison of supervised and unsupervised simplification models on the ASSET test set. PA and DL refer to paraphrasing and deletion, respectively. RM and EX refer to the removal and extraction sub-operations, the sub-operations of the deletion operation. ↑ denotes the higher value, the better. ↓ denotes the lower value, the better.

Value	SARI ↑	Add ↑	Delete ↑	Keep ↑	FKGL ↓	Len
Effect of Meaning Preservation Threshold (H_{mp})						
0.25	38.18	2.15	84.64	27.76	0.42	7.68
0.5	39.49	2.30	83.58	32.59	1.63	8.99
0.6	39.78	2.59	81.94	34.80	2.46	10.29
0.7	39.99	3.16	79.81	36.99	3.27	12.16
Effect of the Removal Threshold of Deletion Operation (t_{dl-rm})						
0.9	37.33	1.93	82.72	27.34	2.19	8.58
1.0	38.03	2.20	81.63	30.25	2.53	9.80
1.1	40.01	3.06	80.43	36.53	3.20	11.72
1.2	39.98	3.15	79.96	36.85	3.26	12.07
Effect of the Paraphrasing Threshold (t_{par})						
0.8	39.99	3.16	79.81	36.99	3.27	12.16
0.9	40.01	3.15	79.55	37.31	3.32	12.24
1.0	39.42	2.69	75.03	40.54	3.84	12.99
1.1	38.55	1.97	70.47	43.23	4.11	13.50
Effect of the Linguistic Acceptability Threshold (H_{la})						
0.6	39.42	3.06	78.19	37.00	3.65	12.54
0.7	39.52	3.00	77.98	37.57	3.68	12.67
0.8	39.69	3.08	77.60	38.40	3.79	12.85
0.9	38.41	2.93	76.87	38.42	4.04	13.04

Table 3: Impact of paraphrasing, deletion, meaning preservation, and linguistic acceptability thresholds on the Newsela dataset.

together (GRS: PA + DL(RM+EX)) gives the best performance on the SARI metric. On the Newsela dataset, our best model outperforms previous unsupervised methods and achieves +1.6 SARI improvement. It also outperforms all supervised methods except Martin et al. (2020b).

For ASSET, even though Martin et al. (2020b) perform better than our best model, we improve the performance over Kumar et al. (2020) by +3.6 SARI points and close the gap between revision-based and generative approaches. Compared to supervised models, our unsupervised model again outperforms others except Martin et al. (2020b) and Omelianchuk et al. (2021). For the ASSET dataset, we observe that our model with only paraphrasing (GRS (PA)) has the best SARI score.

Analyzing the results, we observe that simplification is done differently by human annotators in Newsela than in ASSET. In Newsela, removal of peripheral information through content deletion happens more aggressively. The average reference sentence length is 12.75 compared to 23.04 for the source sentences. However, in ASSET, content removal is conservative and can be handled by paraphrasing alone. The average reference sentence length is 16.54 compared to 19.72 for the source sentences. Simplifications in ASSET focus on lexical simplification, sentence splitting and word reordering.

Martin et al. (2020b) leverage a pretrained BART model (Lewis et al., 2020) and fine-tune it on a paraphrasing dataset containing 1.1 million sequence pairs. Unlike traditional paraphrasing datasets that are structured at the sentence level, their paraphrasing dataset contains multiple sentences in a sequence, thus allowing the model to learn a sentence splitting operation as well. Thus, they outperform

CCD-module	Acc ↑	Rec ↑	Prec ↑	F1 ↑
LS-CCD	84.67	37.36	70.51	48.84
Att-Cls	84.58	47.95	72.59	57.75

Table 4: Performance of different Complex Component Detectors (CCD) on the Complex Word Identification (CWI) task. CWIG3G2 dataset has been used for this evaluation. LS-CCD and Att-Cls refer to the CCD module obtained from Lexical Simplification edit operation of Kumar et al. (2020) and the original CCD module used in GRS design explained in Section 3.3, respectively. ↑ denotes the higher value, the better.

the previous best unsupervised models on ASSET. On Newsela, both GRS and the model from Kumar et al. (2020) perform better than Martin et al. (2020b) since they include an explicit removal edit operation. Martin et al. (2020b) instead do not explicitly perform content removal and only do content deletion by way of paraphrasing. Finally, Kumar et al. (2020) does not perform well on ASSET since they do not perform paraphrasing. Our new design thus combines the advantages of both revision-based and generative approaches.

4.6 Controllability

By manipulating the thresholds for the components of the score function and the edit operations, we can control the amount of deletion, paraphrasing, and the trade-off between simplicity and meaning preservation. We show the results in Table 3 using the GRS (PA + DL) model and the Newsela test set. The column labels have the same meaning as in Tables 1 and 2.

Meaning Preservation Threshold: As mentioned in Section 3.5, meaning preservation is a hard filter in our score function. As the meaning preservation threshold increases, candidate sentences less similar to the original sentence are pruned. Sentences more similar to the original sentence have higher Keep and lower Delete SARI scores. The SARI Add score increases since paraphrasing is prioritized over deletion. Finally, the length of the output sentences increases since the model becomes more conservative.

Removal Threshold of Deletion Operation: By increasing this threshold, the SARI Keep score increases and the SARI Delete score decreases, which also results in increased average length. The SARI Add score increases as well since a higher deletion threshold makes the model conservative on deletions and thus candidates from the paraphrasing operation are more likely to be selected.

Paraphrasing Threshold: Reducing this threshold results in more aggressive paraphrasing. Thus, we observe an increase in the SARI Delete and Add scores since paraphrasing replaces complex words and phrases with simpler ones.

Linguistic Acceptability Threshold: Like meaning preservation, linguistic acceptability is a hard filter in our score function (Section 3.5). As the linguistic acceptability threshold increases, more candidate sentences receive a zero score. This results in a more conservative model that makes fewer changes to the input sentences because the original sentences are already linguistically acceptable. By increasing the linguistic acceptability threshold, the SARI Deletion score drops and the SARI Keep score increases. Also, this results in longer sentences.

4.7 Complex Component Detector Evaluation

To show the effectiveness of the proposed Complex Component Detector (CCD) mentioned in Section 3.3, we evaluate the CCD module on the Complex Word Identification (CWI) task. The task is defined as a sequence tagging problem in which each word in a sentence is tagged as complex or not complex. We use the test set of CWIG3G2 (Yimam et al., 2017), a professionally written news dataset. As explained in Section 3.3, the CCD module used in GRS (denoted by Att-Cls) operates by interpreting the attention matrix of the second layer of the complex-simple classifier. We also compare with the lexical simplification operation of Kumar et al. (2020), denoted by LS-CCD. It uses the IDF scores to find complex words in a sentence.

Table 4 shows the complex word identification performance of the two CCD modules on CWIG3G2. Att-Cls outperforms LS-CCD in recall, precision, and the F1 score. Tables 5 demonstrates the performance of GRS with different CCD modules on ASSET (Alva-Manchego et al., 2020) and Newsela (Xu et al., 2015) test sets. On both datasets, the GRS model using Att-Cls has higher deletion and addition scores compared to the GRS model using LS-CCD. The overall SARI score is considerably higher when using Att-Cls on ASSET.

4.8 Simplification Search Analysis

GRS is an interpretable unsupervised simplification method in which we can trace the simplification process. That is, we know which edit operation is applied on a given complex sentence in each

Model	SARI ↑	Add ↑	Delete ↑	Keep ↑	FKGL ↓	Len
ASSET						
Identity Baseline	20.73	0.00	0.00	62.20	10.02	19.72
Gold Reference	44.89	10.17	58.76	65.73	6.49	16.54
GRS(PA, CCD:LS-CCD)	37.80	5.59	57.39	50.44	7.17	18.75
GRS(PA, CCD:Att-Cls)	**40.41**	**7.00**	**62.37**	**51.88**	**6.70**	17.94
Newsela						
Identity Baseline	12.24	0.00	0.00	36.72	8.82	23.04
GRS(PA+DL(RM), CCD:LS-CCD)	39.30	2.87	78.18	**36.85**	**3.39**	13.54
GRS(PA+DL(RM), CCD:Att-Cls)	**39.61**	**3.18**	**79.13**	36.52	3.45	13.43

Table 5: Comparison of GRS versions that use different Complex Component Detectors (CCD) on ASSET and Newsela. PA and DL refer to paraphrasing and deletion, respectively. RM refers to removal, which is the sub-operation used in deletion operation. ↑ denotes the higher value, the better. ↓ denotes the lower value, the better.

Model	Iterations/Sent (all-Operations)	PA-iterations	DL-iterations	
			RM	EX
Newsela				
GRS: PA	4.72	4.72	–	–
GRS: DL	0.79	–	0.46	0.33
GRS: PA + DL	4.40	3.72	0.37	0.31
ASSET				
GRS: PA	4.18	4.18	–	–
GRS: DL	1.05	–	0.54	0.51
GRS: PA + DL	3.79	2.94	0.39	0.45

Table 6: Analysis of edit operation used during simplification search, showing the average number of simplification iterations of GRS and the average share of each edit operation. PA and DL refer to paraphrasing and deletion, respectively. RM and EX refer to the removal and extraction sub-operations of the deletion operation.

iteration. Table 6 demonstrates how many simplification iterations were needed to simplify a complex sentence, on average, in the Newsela and ASSET datasets. We also show the average frequency of each operation to simplify a given sentence.

Table 6 illustrates that when both edit operations are allowed (GRS:PA+DL), almost four simplification iterations are applied to a sentence, and paraphrasing is generally more common than deletion.

4.9 Human Evaluation

We selected 30 sentences from the ASSET test set for human evaluation. Following (Kriz et al., 2019), we measure Fluency (whether the sentence is grammatical and well-formed), Simplicity (whether it is simpler than the complex sentence), and Adequacy (whether it keeps the meaning of the complex sentence). We asked four volunteers to assess the sentences based on these metrics and evaluate the performance of various models, including GRS. Results are shown in Table 7. All models are unsupervised except Dress-Ls (Zhang and Lapata, 2017).

CCD-module	Adequacy ↑	Simplicity ↑	Fluency ↑	Average ↑
Reference	4.29	4.08	4.76	4.37
GRS(PA)	**3.98**	4.04	4.47	4.17
(Surya et al., 2019)	3.57	3.48	4.32	3.79
(Zhao et al., 2020)	3.89	3.27	4.54	3.89
(Martin et al., 2020b)	3.95	**4.17**	**4.78**	**4.30**
(Kumar et al., 2020)	3.15	3.56	4.15	3.62
(Zhang and Lapata, 2017)	3.67	3.64	4.69	4.00

Table 7: Human evaluation on the ASSET dataset. Adequacy, simplicity, and fluency are human evaluation metrics, and in the fourth column, the average of these metrics is shown. Each row represents a simplification model. Human evaluation scores are based on a 1–5 Likert scale. ↑ denotes the higher value, the better.

The fourth column in Table 7 shows the average score of all three metrics used in the human evaluation. According to the average scores, MUSS (Martin et al., 2020b) has the best performance, followed by GRS. The human evaluation demonstrates that the automatic evaluation (SARI scores shown in Table 2) is aligned with human evaluation scores. GRS has the best performance in meaning preservation (Adequacy). This may be because we have a relatively conservative paraphrasing model. Also, GRS evaluated in this study is only leveraging paraphrasing, and this version is the most conservative.

5 Conclusion

We proposed GRS, a controllable and interpretable method for unsupervised text simplification that bridges the gap between previous unsupervised generative and revision-based approaches. We combined the two approaches by incorporating an explicit paraphrasing edit operation into an iterative simplification search algorithm. Empirically, we showed that GRS has the advantages of both approaches. GRS outperformed state-of-the-art unsupervised methods on the Newsela dataset and reduced the gap between generative and revision-based unsupervised models on the ASSET dataset.

References

Sweta Agrawal, Weijia Xu, and Marine Carpuat. 2021. A non-autoregressive edit-based approach to controllable text simplification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3757–3769, Online. Association for Computational Linguistics.

Fernando Alva-Manchego, Joachim Bingel, Gustavo Paetzold, Carolina Scarton, and Lucia Specia. 2017. Learning how to simplify from explicit labeling of complex-simplified text pairs. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 295–305, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 2020. ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4668–4679, Online. Association for Computational Linguistics.

Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. 2019. EASSE: Easier automatic sentence simplification evaluation. 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): System Demonstrations*, pages 49–54, Hong Kong, China. Association for Computational Linguistics.

John Carroll, Guido Minnen, Yvonne Canning, Siobhan Devlin, and John Tait. 1998. Practical simplification of english newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10. Citeseer.

R. Chandrasekar, Christine Doran, and B. Srinivas. 1996. Motivations and methods for text simplification. In *COLING 1996 Volume 2: The 16th International Conference on Computational Linguistics*.

R Chandrasekar and B Srinivas. 1997. Automatic induction of rules for text simplification1revised version of the article originally published in knowledge-based computer systems: Research and applications. (eds k.s.r. anjaneyulu, m. sasikumar and s. ramani) narosa publishing house, new delhi, 1997.1. *Knowledge-Based Systems*, 10(3):183–190.

Will Coster and David Kauchak. 2011. Learning to simplify sentences using Wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9, Portland, Oregon. Association for Computational Linguistics.

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

Yue Dong, Zichao Li, Mehdi Rezagholizadeh, and Jackie Chi Kit Cheung. 2019. EditNTS: An neural programmer-interpreter model for sentence simplification through explicit editing. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3393–3402, Florence, Italy. Association for Computational Linguistics.

Richard Evans, Constantin Orăsan, and Iustin Dornescu. 2014. An evaluation of syntactic simplification rules for people with autism. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 131–140, Gothenburg, Sweden. Association for Computational Linguistics.

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic multi-level multi-task learning for sentence simplification. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 462–476, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*.

Chris Hokamp and Qun Liu. 2017. Lexically constrained decoding for sequence generation using grid beam search. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1535–1546, Vancouver, Canada. Association for Computational Linguistics.

J. Edward Hu, Huda Khayrallah, Ryan Culkin, Patrick Xia, Tongfei Chen, Matt Post, and Benjamin Van Durme. 2019a. Improved lexically constrained decoding for translation and monolingual rewriting. 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 839–850, Minneapolis, Minnesota. Association for Computational Linguistics.

J. Edward Hu, Abhinav Singh, Nils Holzenberger, Matt Post, and Benjamin Van Durme. 2019b. Large-scale, diverse, paraphrastic bitexts via sampling and clustering. In *Proceedings of the 23rd Conference on Computational Natural Language Learning (CoNLL)*, pages 44–54, Hong Kong, China. Association for Computational Linguistics.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural CRF model for

sentence alignment in text simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7943–7960, Online. Association for Computational Linguistics.

Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. In *OTM Confederated International Conferences" On the Move to Meaningful Internet Systems"*, pages 735–747. Springer.

Reno Kriz, João Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. 2019. Complexity-weighted loss and diverse reranking for sentence simplification. 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 3137–3147.

Dhruv Kumar, Lili Mou, Lukasz Golab, and Olga Vechtomova. 2020. Iterative edit-based unsupervised sentence simplification. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7918–7928, Online. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

Mounica Maddela, Fernando Alva-Manchego, and Wei Xu. 2021. Controllable text simplification with explicit paraphrasing. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3536–3553, Online. Association for Computational Linguistics.

Louis Martin, Éric de la Clergerie, Benoît Sagot, and Antoine Bordes. 2020a. Controllable sentence simplification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4689–4698, Marseille, France. European Language Resources Association.

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

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 435–445, Baltimore, Maryland. Association for Computational Linguistics.

Shashi Narayan and Claire Gardent. 2016. Unsupervised sentence simplification using deep semantics. In *Proceedings of the 9th International Natural Language Generation conference (INLG)*, pages 111–120.

Daiki Nishihara, Tomoyuki Kajiwara, and Yuki Arase. 2019. Controllable text simplification with lexical constraint loss. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, pages 260–266, Florence, Italy. Association for Computational Linguistics.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 85–91, Vancouver, Canada. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vipul Raheja, and Oleksandr Skurzhanskyi. 2021. Text Simplification by Tagging. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 11–25, Online. Association for Computational Linguistics.

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.

Matt Post and David Vilar. 2018. Fast lexically constrained decoding with dynamic beam allocation for neural machine translation. 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 1314–1324, New Orleans, Louisiana. Association for Computational Linguistics.

Machel Reid and Victor Zhong. 2021. LEWIS: Levenshtein editing for unsupervised text style transfer. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3932–3944, Online. Association for Computational Linguistics.

Nils Reimers and Iryna Gurevych. 2019. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics.

Carolina Scarton and Lucia Specia. 2018. Learning simplifications for specific target audiences. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*,

pages 712–718, Melbourne, Australia. Association for Computational Linguistics.

Sanja Štajner and Maja Popovic. 2016. Can text simplification help machine translation? In *Proceedings of the 19th Annual Conference of the European Association for Machine Translation*, pages 230–242.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018. Semantic structural evaluation for text simplification. 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 685–696, New Orleans, Louisiana. Association for Computational Linguistics.

Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. 2019. Unsupervised neural text simplification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2058–2068, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

David Vickrey and Daphne Koller. 2008. Sentence simplification for semantic role labeling. In *Proceedings of ACL-08: HLT*, pages 344–352, Columbus, Ohio. Association for Computational Linguistics.

Jesse Vig. 2019. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, Florence, Italy. Association for Computational Linguistics.

Alex Warstadt, Amanpreet Singh, and Samuel R. Bowman. 2019. Neural network acceptability judgments. *Transactions of the Association for Computational Linguistics*, 7:625–641.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1015–1024, Jeju Island, Korea. Association for Computational Linguistics.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Seid Muhie Yimam, Sanja Štajner, Martin Riedl, and Chris Biemann. 2017. CWIG3G2 - complex word identification task across three text genres and two user groups. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 401–407, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 584–594, Copenhagen, Denmark. Association for Computational Linguistics.

Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018. Integrating transformer and paraphrase rules for sentence simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3164–3173, Brussels, Belgium. Association for Computational Linguistics.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. *Proceedings of the AAAI Conference on Artificial Intelligence*, 34(05):9668–9675.

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of the 23rd International Conference on Computational Linguistics (Coling 2010)*, pages 1353–1361, Beijing, China. Coling 2010 Organizing Committee.