Text Simplification by Tagging

Kostiantyn Omelianchuk[*] **Vipul Raheja**[*] **Oleksandr Skurzhanskyi**[*]
Grammarly
`firstname.lastname@grammarly.com`

Abstract

Edit-based approaches have recently shown promising results on multiple monolingual sequence transduction tasks. In contrast to conventional sequence-to-sequence (Seq2Seq) models, which learn to generate text from scratch as they are trained on parallel corpora, these methods have proven to be much more effective since they are able to learn to make fast and accurate transformations while leveraging powerful pre-trained language models. Inspired by these ideas, we present TST, a simple and efficient **T**ext **S**implification system based on sequence **T**agging, leveraging pre-trained Transformer-based encoders. Our system makes simplistic data augmentations and tweaks in training and inference on a pre-existing system, which makes it less reliant on large amounts of parallel training data, provides more control over the outputs and enables faster inference speeds. Our best model achieves near state-of-the-art performance on benchmark test datasets for the task. Since it is fully non-autoregressive, it achieves faster inference speeds by over 11 times than the current state-of-the-art text simplification system.

1 Introduction

Text Simplification is the task of rewriting text into a form that is easier to read and understand while preserving its underlying meaning and information. It has been shown to be valuable in providing assistance in terms of readability and understandability to children (Belder and Moens, 2010; Kajiwara et al., 2013), people with language disabilities like aphasia (Carroll et al., 1998, 1999; Devlin and Unthank, 2006), dyslexia (Rello et al., 2013a,b), or autism (Evans et al., 2014); non-native English speakers (Petersen and Ostendorf, 2007; Paetzold, 2015; Paetzold and Specia, 2016a,b; Pellow and Eskenazi, 2014), and people with low literacy skills or reading ages (Max, 2006; Aluísio et al., 2008; Gasperin et al., 2009; Watanabe et al., 2009). Moreover, it has also been successfully leveraged as a pre-processing step to improve the performance of various NLP tasks such as parsing (Chandrasekar et al., 1996), summarization (Beigman Klebanov et al., 2004; Silveira and Branco, 2012), semantic role labeling (Vickrey and Koller, 2008; Woodsend and Lapata, 2017) and machine translation (Gerber and Hovy, 1998; Štajner and Popovic, 2016; Hasler et al., 2017).

Evolving from the approaches ranging from building hand-crafted rules (Chandrasekar et al., 1996; Siddharthan, 2006) to syntactic and lexical simplification via synonyms and paraphrases (Siddharthan, 2014; Kaji et al., 2002; Horn et al., 2014; Glavaš and Štajner, 2015), the task has gained popularity as a monolingual Machine Translation (MT) problem, where the system learns to "translate" a given complex sentence to its simplified form. Initially, Statistical phrase-based (SMT) and Syntactic-based Machine Translation (SBMT) techniques (Zhu et al., 2010; Specia, 2010; Coster and Kauchak, 2011; Wubben et al., 2012; Narayan and Gardent, 2014; Štajner et al., 2015; Xu et al., 2016a) were successfully applied as a way to learn simplification rewrites implicitly from complex-simple sentence pairs, often in combination with hand-crafted rules or features. More recently, several Neural Machine Translation-based (NMT) systems have been developed with promising results (Sutskever et al., 2014; Cho et al., 2014; Bahdanau et al., 2015), and their successful application to text simplification, either in combination with SMT or other data-driven approaches (Zhang et al., 2017; Zhao et al., 2018b); or strictly as neural models (Wang et al., 2016; Nisioi et al., 2017; Zhang and Lapata, 2017; Štajner and Nisioi, 2018; Guo et al., 2018; Vu et al., 2018; Li et al., 2018; Kriz et al., 2019; Surya et al., 2019; Zhao et al., 2020a), has emerged as the state-of-the-art.

[*]Authors contributed equally to this work; names are given in alphabetical order.

Human editors perform several rewriting transformations in order to simplify a sentence, such as lexical paraphrasing, changing the syntactic structure, or removing superfluous information from the sentence (Petersen and Ostendorf, 2007; Aluísio et al., 2008; Mallinson et al., 2020). Therefore, even though NMT-based sequence-to-sequence (Seq2Seq) approaches offer a generic framework for modeling almost any kind of sequence transduction, target texts in these approaches are typically generated from scratch - a process which can be unnecessary for monolingual editing tasks such as text simplification, owing to these aforementioned transformations. Moreover, these approaches have a few shortcomings that make them inconvenient for real-world deployment. First, they give limited insight into the simplification operations and provide little control or adaptability to different aspects of simplification (e.g., lexical vs. syntactical simplification). This inhibits interpretability and explainability, which is crucial for real-world settings. Second, they are not sample-efficient and require a large number of complex-simple aligned sentence pairs for training, which requires considerable human effort to obtain. Third, these models typically employ an autoregressive decoder, i.e., output texts are generated in a sequential, non-parallel fashion, and hence, are generally characterized by slow inference speeds.

Based on the aforementioned observations and issues, text-editing approaches have recently regained significant interest (Gu et al., 2019; Dong et al., 2019; Awasthi et al., 2019; Malmi et al., 2019; Omelianchuk et al., 2020; Mallinson et al., 2020). Typically, the set of edit operations in such tasks is fixed and predefined ahead of time, which on one hand limits the flexibility of the model to reconstruct arbitrary output texts from their inputs, but on the other, leads to higher sample-efficiency as the limited set of allowed operations significantly reduces the search space (Mallinson et al., 2020). This pattern is especially true for monolingual settings where input and output texts have relatively high degrees of overlap. In such cases, a natural approach is to cast the task of conditional text generation into a text-editing task, where the model learns to reconstruct target texts by applying a set of edit operations to the inputs. We leverage this insight in our work, and simplify the task from sequence generation or editing, going a step further, to formulate it as a sequence tagging task. In addition to being sample efficient, thanks to the separation of various edit operations in the form of tags, the system has better interpretability and explainability. Finally, since for sequence tagging we don't need to predict tokens one-by-one as in autoregressive decoders, the inference is naturally parallelizable and therefore runs many times faster.

Following from the success of the aforementioned monolingual edit-tag based systems, we propose to leverage the current state-of-the-art model for Grammatical Error Correction by Omelianchuk et al. (2020) (GECToR) and adapt it to the task of Text Simplification. In summary, we make the following contributions:

- We develop a Text Simplification system by adapting the GECToR model to Text Simplification, leveraging Transformer-based encoders trained on large amounts of human-annotated and synthetic data.[1] Empirical results demonstrate that our system achieves near state-of-the-art performance on benchmark test datasets in terms of readability and simplification metrics.
- We propose crucial data augmentations and tweaks in training and inference and show their significant impact on the task: enabling the model to learn to edit the sentences more effectively, rather than relying heavily on copying the source sentences, leading to a higher quality of simplifications.
- Since our model is a non-autoregressive sequence tagging model, it achieves over 11 times speedup in inference time, compared to the state-of-the-art for Text Simplification.

2 Related Work

Recent text editing works have shown promising results of reformulating multiple monolingual sequence transduction tasks into sequence tagging tasks compared to the conventional Seq2Seq sequence generation formulation. This observation is especially true for tasks where input and output sequences have a large overlap. Generally, these works try to simplify monolingual sequence transduction by explicitly modeling edit operations such as KEEP, ADD/INSERT and DELETE. Alva-Manchego et al. (2017) proposed the first such formulation, employing a BiLSTM to predict edit

[1] Available at `https://github.com/grammarly/gector#text-simplification`

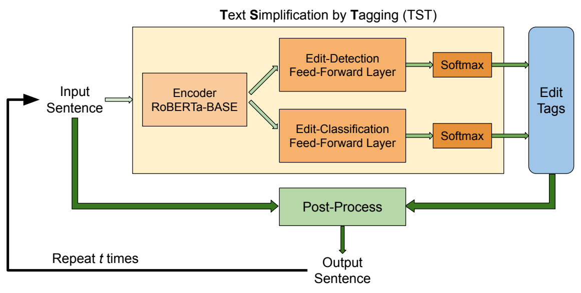
Text Simplification by Tagging (TST)
Input Sentence
Encoder RoBERTa-BASE
Edit-Detection Feed-Forward Layer
Edit-Classification Feed-Forward Layer
Softmax
Softmax
Edit Tags
Post-Process
Output Sentence
Repeat *t* times

Figure 1: **T**ext **S**implification by **T**agging (TST): A given sentence undergoes multiple iterations of tag-and-edit transformations, where, in each iteration, it is tagged using custom token-level edit-tags, and the sequence of tags is converted back to text by applying those edits, iteratively making simplifying edits.

labels sequentially. Our model for sentence simplification does not rely on external simplification rules nor alignment tools. Ribeiro et al. (2018) proposed an approach applied only to character deletion and insertion and was based on simple patterns. LaserTagger (Malmi et al., 2019) combines a BERT encoder with an autoregressive Transformer decoder to similarly predict the aforementioned three main edit operations for several text editing tasks. In contrast, in our system, the decoder is a softmax layer. Similarly, EditNTS (Dong et al., 2019) and PIE (Awasthi et al., 2019) predict edit labels, developed specifically for text simplification and GEC, respectively. While EditNTS employs an autoregressive encoder-decoder based neural programmer-interpreter model, PIE differs from our work because of our custom edit transformations and incorporation of a pre-trained Transformer encoder for sequence tagging. Levenshtein Transformer (Gu et al., 2019), an autoregressive model that performs text editing by executing a sequence of deletion and insertion actions, is another recent work along similar lines. More recently, Mallinson et al. (2020) proposed Felix - a text-editing-based system for multiple generation tasks, splitting the text-editing task into two subtasks: tagging and insertion. Their tagging model employs a Pointer mechanism, while the insertion model is based on a Masked Language Model.

3 System Description

Following recent works such as Malmi et al. (2019); Awasthi et al. (2019); Omelianchuk et al. (2020), who leveraged similar frameworks for different text editing problems such as GEC, Sentence Fusion, and Abstractive Summarization, we formulate the task of Text Simplification as a tagging problem.

Specifically, our system is based on GECToR (Omelianchuk et al., 2020), an iterative sequence-tagging system that works by predicting token-level edit operations, originally developed for Grammatical Error Correction (GEC). We adapt the GECToR framework for the task of Text Simplification, with minimal modifications to the original architecture. Our system consists of three main parts: (a) defining the custom transformations (token-level edit-tags), (b) performing iterative sequence tagging to convert target sequences to tag sequences, (c) fine-tuning of pre-trained Transformers to predict the tag sequences. Each of these components are described below.

3.1 Edit Transformations

In order to formulate the task as a tagging problem, building on the aforementioned edit-tagging-based approaches, we use custom token-level edit operations (also referred to as **edit-tags** or **transformations**) to perform text simplification. Formally, given a sentence x: $[x_1, x_2, \ldots, x_N]$, and its simplified form y: $[y_1, y_2, \ldots, y_M]$ as the target sentence, we aim to predict an edit tag $t_i \in \tau$ (τ denoting the edit-tag vocabulary) for each token x_i in x, generating a sequence of edit-tags of the same length N as the input sequence x, such that $t_i(x_i)$: applying the edit operation represented by the edit-tag t_i to the input token x_i at each position i, reconstructs the target sequence y, even though $M \leq N$.

We reuse the edit transformations in GECToR, which were developed for GEC. We chose to do so because we found a significantly high overlap of 92.64% in the tag distributions between the GEC and Text Simplification domains. This was done by building the edit-tag vocabularies independently on both (GEC and Text Simplification) datasets and comparing the tag distributions represented by the two vocabularies. This was not surprising since these edit-tags have been obtained from huge amounts of synthetic GEC data, they are expected to have good coverage with many standard monolingual text editing problems. Additionally, using the same edit-tags is a necessary pre-requisite to leverage GEC initialization in the model (Section 4.3), which we later show to be quite impactful for our text simplification system (Section 6.1). Consequently, the edit space τ is of size 5000, out of which 4971 are basic edit-tags and 29 are token-independent GEC-specific edit-tags (such as $TRANSFORM_VERB_VB_VBZ, which converts a verb in its base form to its third person singular present tense form). Further, the aforementioned 4971 basic edit-tags are made up of token-independent KEEP and DELETE tags (which simply keep or delete the given word(s) on which they are applied), 1167 token-dependent APPEND tags (such as $APPEND_just, which appends the word "just" to the given word) and 3802 token-dependent REPLACE tags (such as $REPLACE_really, which replaces the given word with the word "really").

3.2 Iterative Sequence Tagging

As described in Section 3.1, we predict the edit-tags t_i for each input token x_i in the source sequence x. These predicted tag-encoded transformations are then applied to the source sentence to get the simplified sentence. Since some simplification operations in a sentence may depend on others, applying the sequence tagger only once may not be enough to fully generate the simplified form of a given sentence. Accordingly, we use the iterative correction approach from Awasthi et al. (2019) and Omelianchuk et al. (2020), and use the sequence tagger to tag the now modified sequence, and apply the corresponding transformations on the new edit-tags, which changes the sentence further. We repeat this process for a fixed number of iterations, which can be adjusted to trade off qualitative performance for improved speed. In our framework, we experimented between 1-5 iterations.

3.3 Tagging Model

We use the GECToR sequence tagging model with a pre-trained RoBERTa$_{\text{BASE}}$ Transformer (Liu et al., 2019) as the encoder, stacked with two concurrent feed-forward layers, followed by corresponding Softmax layers. Owing to our choice of encoder, we use Byte-Pair Encoding (BPE) (Sennrich et al., 2016) as our tokenization technique.

As shown in Fig. 1, these feed-forward layers are responsible for detecting and classifying edits, respectively. For every position in the input sequence, the edit-detection layer predicts the probability an edit exists, whereas the edit-classification layer predicts the type of edit-tag. The edit-tag sequence generated as the output of the edit-classification layer is gated by the output of the edit-detection layer. i.e. if the output of the edit-detection layer is below the *minimum edit probability* threshold (described in Section 4.6) at any position in the predicted sequence, we do not make any edits.

4 Experimental Setup

4.1 Data Sources

We use **WikiSmall** and **WikiLarge**, two benchmark datasets for the text simplification task[2], for our experiments. These datasets were constructed from automatically-aligned complex-simple sentence pairs from English Wikipedia (EW) and Simple English Wikipedia (SEW). WikiSmall (Zhu et al., 2010) contains one reference simplification per sentence. We use the standardized split of this dataset released by Zhang and Lapata (2017), with 88*k* instances for training, 205 for validation and the same original 100 instances for testing. WikiLarge is a larger set of similarly automatically-aligned complex-simple sentence pairs, compiled from previous extractions of EW-SEW and WikiSmall (Zhu et al., 2010; Woodsend and Lapata, 2011; Kauchak, 2013). Similar to WikiSmall, we use the training set for this dataset provided by Zhang and Lapata (2017) consisting of 296*k* sentence pairs. For simplicity, we refer to this training data (WikiSmall + WikiLarge) as **WikiAll**.

For validation and test sets, we use the Turkcorpus (Xu et al., 2016a) and ASSET (Alva-Manchego et al., 2020) datasets, which were both created from WikiLarge using the same 2000 validation and 359 test source sentences, where each complex sentence consists of multiple crowd-sourced reference simplifications. Specifically, Turkcorpus contains 8 reference simplifications, and ASSET contains 10 references per source sentence.

Table 1 provides other statistics on these datasets.

4.2 Data pre-processing

WikiAll data contains special tokens to represent parentheses (symbolized by -LRB- and -RRB-) from prior tokenizations. We heuristically decide to remove these tokens (and any tokens between them) from both source and target sentences. Doing this led to consistent improvements in all our experiments, described further in Section 6.2. Additionally, for tokenization, we use the HuggingFace Tokenizers[3] Python library to tokenize the whole

[2]Another widely-used dataset for the task, the Newsela Corpus (Xu et al., 2015), could not be used due to its extremely rigid legal and licensing requirements.

[3]https://github.com/huggingface/tokenizers

Split	Dataset		Sentences	Tokens
Train	WikiAll	WikiSmall	88k	3.9M
		WikiLarge	296k	11.7M
	WikiBT	WikiLarge (En-De)	293k	11.2M
		WikiLarge (En-Fr)	293k	11.5M
Dev	WikiSmall		205	9.5k
	WikiLarge	TurkCorpus	2000	75k
		ASSET	2000	72k
Test	WikiSmall		100	5k
	WikiLarge	TurkCorpus	359	15.8k
		ASSET	359	14.1k

Table 1: Dataset splits and sizes.

sentence (as opposed to the approach in GECToR which tokenized each word in the sentence separately). This change led faster and more accurate tokenization as the one originally used in RoBERTa.

4.3 Pre-Training

For our experiments, we use two versions of the tagging model described in Section 3.3. The first version is a pre-trained RoBERTa$_{\text{BASE}}$ encoder with randomly initialized feed-forward layers. We refer to this model as **TST-BASE** (**T**ext **S**implification by **T**agging - Baseline). The second version of the model is a TST-BASE model fine-tuned on the Grammatical Error Correction (GEC) task: henceforth denoted as **TST-GEC**.[4]

4.4 Data Augmentation

We hypothesize that our text simplification models can benefit from an increase of the training data, and experimentally confirm this by training and evaluating our models with additional training data. We generate synthetic training data from the source sentences of WikiAll. We used two approaches to do so: back-translation and ensemble distillation, described below.

4.4.1 Back-Translation

We use the Transformer-based NMT models trained by Tiedemann and Thottingal (2020) to generate the back-translated versions of the target side of the parallel WikiAll data. These models were trained on OPUS data[5] using Marian-NMT[6] and released as part of the HuggingFace Transformers Library (Wolf et al., 2019). All models are Transformer encoder-decoders with 6 layers in each component. Specifically, we used the bilingual EN-FR, FR-EN, EN-DE and DE-EN models[7] to translate WikiAll data from (a) English to French, and back to English, and (b) English to German and back to English. Doing so tripled the amount of WikiAll data available for training (Table 1). The backtranslated WikiAll data is henceforth collectively referred to as **WikiBT**.

4.4.2 Ensemble Distillation

We leverage knowledge distillation on ensemble teacher models (Freitag et al., 2017) to augment our training data. We first create an ensemble teacher model by training models on WikiAll and WikiBT data. Specifically, for building the teacher ensemble, we first train the following constituent TST models:

1. TST: Trained on WikiAll
2. TST-GEC: Trained on WikiAll
3. TST: Trained on WikiAll + WikiBT

The predictions of the ensemble are computed by taking the `argmax` of the averaged class-wise probabilities of the constituent models at every token. We get the predictions from this ensemble consisting of the aforementioned three constituent models on WikiAll data. In this way, we produce new references for the training data which can be used by our final model (referred to as the student network) to simulate the teacher network ensemble. We then combine this ensemble-generated training data (hereby referred to as **WikiEns**) together with the original WikiAll data, doubling the amount of training data. Our final model (the student network, denoted henceforth as **TST-FINAL**) is then trained on this combined WikiEns + WikiAll dataset. It is worth noting that the student and the constituent teacher models have exactly the same architecture.

4.5 Training

We train our models with AllenNLP and Transformers. Our baseline (TST-BASE) mostly follows the settings in Omelianchuk et al. (2020). We train the model for 50 epochs, where we freeze the encoder weights during the first two epochs of training. We use Adam optimizer (Kingma and Ba, 2015), where the learning rate starts from 1e-5 and reduces by a factor of 0.1 when the validation loss has stopped

[4]We refer the reader to Omelianchuk et al. (2020) for details on training the model for GEC.

[5]http://opus.nlpl.eu/

[6] https://marian-nmt.github.io/

[7]https://huggingface.co/Helsinki-NLP/opus-mt-<L1>-<L2>

improving for 10 epochs. We perform early stopping after 3 epochs, based on the performance on the validation set. Other training hyper-parameters are listed in Appendix A.

4.6 Inference Tweaks

One of the advantages of edit-tag-based approaches is that they provide greater control over the system output. Building on Omelianchuk et al. (2020), we use *confidence biases* and *minimum edit probability* as additional inference hyper-parameters that we tune to push the model to perform more precise edits.

Specifically, we add confidence biases to the probabilities of KEEP and DELETE edit-tags: responsible for not changing the source token and deleting the source token, respectively. We create these additional hyper-parameters for just these edit-tags because they are the most frequently used edit-tags for the Text Simplification task. Moreover, since they are token-independent, it provides the framework with additional robustness on the task without introducing too many additional hyper-parameters. In this way, we were able to drive the model to keep/delete more tokens if the corresponding confidence bias was positive and to keep/delete fewer tokens if it was negative. We also add a sentence-level minimum edit probability threshold (ϵ) for the output of the edit detection layer. This hyper-parameter enabled the model to predict only the most confident edits. Thus, we were able to increase precision by trading off the recall and achieve better performance.

These hyper-parameters were tuned using a combination of random search and Bayesian search (Nogueira, 2014) on the respective validation sets. Section 6 further describes the impact of the aforementioned tweaks on the system. Final values of these hyper-parameters are listed in Appendix A.

4.7 Evaluation Metrics

We report the results using two widely used metrics in Text Simplification literature: **FKGL** (Kincaid et al., 1975), and **SARI** (Xu et al., 2016b). Prior work has also used **BLEU** (Papineni et al., 2002) as a metric, but recent work has found that it is not a suitable metric for evaluating text simplification, because it was found to be negatively correlated with simplicity, essentially penalizing simpler sentences (Sulem et al., 2018).

FKGL (**F**lesch-**K**incaid **G**rade **L**evel) is used to measure the readability of the generated sentence, where a lower score indicates simpler output. FKGL doesn't use source sentences or references for computing the score. It is a linear combination of the number of words per sentence (system output) and the number of syllables per word. On the other hand, SARI (**S**ystem output **A**gainst **R**eferences and against the **I**nput sentence) evaluates the quality of the output by comparing the generated sentence to a set of reference sentences in terms of correctly inserted, kept and deleted n-grams ($n \in 1, 2, 3, 4$). We report the overall SARI metric, and scores on the three rewrite operations used in SARI: the F1-scores of add (ADD), delete (DELETE) and keep (KEEP) operations. FKGL and SARI are both measured at corpus-level. We computed all the evaluation metrics using the EASSE[8] Python package (Alva-Manchego et al., 2019).

5 Results

5.1 Text Simplification and Readability

Table 2 summarizes the results of our evaluations on TurkCorpus, ASSET and WikiSmall test sets. To ensure robustness of results, we report average scores of 4 runs with different random seeds. We compare the results of our baseline model (TST-BASE) and our final model (TST-FINAL) against recent state-of-the-art Neural Text Simplification models. Additionally, we compare against a reference baseline similar to Martin et al. (2020b), where we compute the scores in a leave-one-out scenario where each reference is evaluated against all the others and then scores are averaged over all references. TST-FINAL consists of all the enhancements mentioned in Section 4 added on top of TST-BASE: data pre-processing, GEC-initialization, data augmentation, and inference tweaks. In terms of the FKGL score, our system achieves better results than the reference baselines on TurkCorpus, and comes within 0.5 points on ASSET and WikiSmall. Compared to the state-of-the-art (Martin et al., 2020b), it improves by 0.23 FKGL points on average, indicating that the simplifying edits made by TST are easier to understand.

In terms of SARI metrics, TST-BASE achieves a competitive score of 39.17 on TurkCorpus, and a state-of-the-art SARI score of 43.11 on WikiSmall, outperforming the previous state-of-the-art result by a huge margin of 6.19 SARI points. This shows that simply using our baseline architecture to train a Text Simplification model on WikiAll can achieve

[8] https://github.com/feralvam/easse

	SARI↑	ADD↑	DELETE↑	KEEP↑	FKGL↓
Recent Works					
Xu et al. (2016b)	39.96	5.96	41.42	72.52	7.29
Nisioi et al. (2017)	35.66	2.99	28.96	**75.02**	8.42
Zhang and Lapata (2017)	37.27	-	-	-	6.62
Alva-Manchego et al. (2017)‡	37.08	2.94	43.20	65.10	**5.35**
Vu et al. (2018)	36.88	-	-	-	-
Zhao et al. (2018a)	40.42	5.72	42.23	73.41	7.79
Guo et al. (2018)	37.45	-	-	-	7.41
Qiang (2018)	37.21	-	-	-	6.56
Surya et al. (2019)	34.96	-	-	-	-
Dong et al. (2019)	38.22	3.36	39.15	72.13	7.3
Zhao et al. (2020b)	37.25	2.87	40.06	68.82	-
Mallinson et al. (2020)	38.13	3.55	40.45	70.39	8.98
Martin et al. (2020a)	41.38	-	-	-	7.29
Martin et al. (2020b)	**42.53**$_{\pm 0.36}$	-	-	-	7.60$_{\pm 1.06}$
Reference Baseline	40.02$_{\pm 0.72}$	6.21$_{\pm 0.60}$	**70.15**$_{\pm 1.35}$	43.69$_{\pm 1.46}$	8.77$_{\pm 0.19}$
Our System					
TST-Base	39.17$_{\pm 0.77}$	3.62$_{\pm 0.41}$	41.61$_{\pm 3.14}$	72.29$_{\pm 1.45}$	8.08$_{\pm 0.31}$
TST-Final	41.46$_{\pm 0.32}$	**6.96**$_{\pm 0.44}$	47.87$_{\pm 0.75}$	69.56$_{\pm 1.19}$	7.87$_{\pm 0.19}$

‡ Quoted from the re-implementation by Dong et al. (2019).

(a) TurkCorpus

	SARI↑	ADD↑	DELETE↑	KEEP↑	FKGL↓
Recent Works					
Martin et al. (2020a)	40.13	-	-	-	7.29
Martin et al. (2020b)	44.15$_{\pm 0.6}$	-	-	-	7.60$_{\pm 1.06}$
Reference Baseline	**44.89**$_{\pm 0.90}$	**10.17**$_{\pm 1.20}$	58.76$_{\pm 2.24}$	**65.73**$_{\pm 2.03}$	**6.49**$_{\pm 0.42}$
Our System					
TST-Base	37.4$_{\pm 1.62}$	3.62$_{\pm 0.59}$	47.22$_{\pm 4.5}$	61.37$_{\pm 0.52}$	8.08$_{\pm 0.31}$
TST-Final	43.21$_{\pm 0.3}$	8.04$_{\pm 0.29}$	**64.25**$_{\pm 1.22}$	57.35$_{\pm 1.68}$	6.87$_{\pm 0.27}$

(b) ASSET

	SARI↑	ADD↑	DELETE↑	KEEP↑	FKGL↓
Recent Works					
Zhang and Lapata (2017)	27.24	-	-	-	7.55
Alva-Manchego et al. (2017)‡	30.50	2.72	76.31	12.46	9.38
Vu et al. (2018)	29.75	-	-	-	-
Guo et al. (2018)	28.24	-	-	-	6.93
Qiang (2018)	26.49	-	-	-	10.75
Dong et al. (2019)	32.35	2.24	**81.30**	13.54	**5.47**
Zhao et al. (2020b)	36.92	2.04	72.79	35.93	-
Reference Baseline	-	-	-	-	8.74
Our System					
TST-Base	43.11$_{\pm 1.87}$	4.66$_{\pm 1.31}$	61.13$_{\pm 4.73}$	**63.54**$_{\pm 2.75}$	8.41$_{\pm 1.01}$
TST-Final	**44.67**$_{\pm 1.26}$	**8.12**$_{\pm 0.92}$	64.87$_{\pm 2.09}$	61.01$_{\pm 1.76}$	9.29$_{\pm 0.9}$

‡ Quoted from the re-implementation by Dong et al. (2019).

(c) WikiSmall

Table 2: Comparison of our system against recent state-of-the-art Neural Text Simplification models on TurkCorpus, ASSET and WikiSmall test sets.

competitive performance on the task. On the other hand, the TST-FINAL makes significant improvements over TST-BASE. On TurkCorpus and ASSET, it comes within 1 SARI point of the current state-of-the-art (Martin et al., 2020b), outperforming all other prior text simplification models in literature. On WikiSmall, it further improves its state-of-the-art performance from TST-BASE to achieve a SARI score of 44.67.

It can be seen that compared to prior works, the most significant improvements in both the TST models come from ADD and DELETE operations. It is noteworthy that TST-FINAL is able to achieve the highest F1 scores on ADD (6.96) and DELETE (47.87) SARI operations reported in literature on TurkCorpus. On the ASSET dataset, the F1 scores on ADD and DELETE operations improve further to 8.04 and 64.25 respectively, improving by large margins over TST-BASE. Similarly, it outperforms the state-of-the-art on ADD operations on WikiSmall. This shows that models proposed in prior works learned a safe, but inefficient strategy of simplification - leaning heavily on copying the sources sentences directly, owing to their high KEEP scores. By contrast, our model learns to edit the sentences better, as shown by the lower rates of keeping the source sentences unchanged. This is further verified by the fact that outputs of prior works[9] have much longer output sentence lengths (avg. 19.26 words) compared to ours (avg. 16.7 words), leading to more effective simplifications.

5.2 Inference Time

We also compare our system's inference times against the current state-of-the-art text simplification systems. Specifically, we compare against ACCESS (trained on WikiLarge data) (Martin et al., 2020a) and BART+ACCESS (Martin et al., 2020b) (trained on WikiLarge + MINED data) systems. We used the publicly available model checkpoint for ACCESS to compare against Martin et al. (2020a). Direct comparison against BART+ACCESS was not possible because of the lack of publicly available code. Therefore, we used BART (Lewis et al., 2020) for text summarization as a proxy for Martin et al. (2020b). We ran all systems with batch size 128 on the TurkCorpus test set 100 times, using NVIDIA Tesla V100. Within a single run, the results were averaged across all batches. We took into account only the actual inference time and omitted any initialization times.

The results in Table 3 show that the inference speeds[10] of TST are at least 6 times faster than ACCESS and 11.75 times faster than pure BART which is the crucial component of the current state-of-the-art (Martin et al., 2020b). The impact of a non-autoregressive model architecture can be clearly seen here since TST is a sequence tagging system and does not need to predict edits one-by-one as done by auto-regressive transformer decoders (like the one used in ACCESS). There-

[9]Measured on TurkCorpus. This information was not available for ASSET and WikiSmall

[10]We compared ACCESS and BART with beam size 8 and TST with 2 iterations, as reported on TurkCorpus

System	Inference time (sec)
BART, beam size = 8	2.82
BART, beam size = 2	1.95
ACCESS, beam size = 8	1.43
ACCESS, beam size = 1	1.14
TST, 5 iterations	0.43
TST, 4 iterations	0.39
TST, 3 iterations	0.33
TST, 2 iterations	0.24
TST, 1 iteration	0.13

Table 3: Average inference time per batch. In the context of TST, iterations refers to the number of iterations mentioned in Section 3.2

fore, the inference is naturally parallelizable and therefore runs many times faster.

6 Ablation Study

In this section, we present ablation experiments for each of the enhancements described in Section 4, and applied to TST-BASE to obtain TST-FINAL. The results of these experiments (Table 4) are reported on SARI and FKGL scores, averaged between ASSET and TurkCorpus test datasets. Each result is reported using an average of 4 runs for each experiment. Overall, the enhancements improve the SARI score by 4.0 points and FKGL by 0.21 points, while reducing variance in both cases.

6.1 GEC Initialization

We improve our strong baseline model TST-BASE by pre-training it on the GEC task.[11] Even though we find that using TST-GEC leads to modest immediate improvements (+0.1 SARI point) compared to TST-BASE, we found that adding other enhancements without the GEC-pre-training were not as effective, with the final model (TST-BASE + Filtering + WikiEns + InfTweaks) achieving an average SARI score of 40.01 - significantly lower than the one with GEC-pre-training (42.3).

These results show that pre-training TST-BASE on GEC is an effective way to initialize the model for Text Simplification, since it equips the model to make additions and deletions, which are then further improved during training on the text simplification data. This was also not unexpected because the edit-tags were obtained from huge amounts of GEC data, and are expected to have good coverage with regards to many standard monolingual text

[11]We refer the reader to Omelianchuk et al. (2020) for details on training the model for GEC.

System	SARI ↑	FKGL ↓
TST	38.3 ± 1.36	8.08 ± 0.31
+ GEC	38.4 ± 0.83	8.32 ± 0.26
+ Filtering	39.1 ± 0.48	7.66 ± 0.25
+ WikiBT	39.5 ± 0.01	7.5 ± 0.06
+ WikiEns (- WikiBT)	40.3 ± 0.15	**7.48** ± **0.2**
+ InfTweaks	**42.3** ± **0.25**	7.87 ± 0.19

Table 4: Average SARI and FKGL scores (ASSET and TurkCorpus test sets)

editing problems - as also observed by a high overlap in the tag distributions between the GEC and Text Simplification domains (Section 3.1).

6.2 Data Pre-processing

As mentioned in Section 4.2, we removed special tokens found in Wikipedia data such as `-LRB-` and `-RRB-`, along with the text enclosed by these tokens, in both source and target sentences. We find that using GEC initialization together with filtering brackets was beneficial to the system (+0.8 SARI points), and also decreased the variance in the results. The benefit of this step is towards improving text simplification quality is also seconded by a significantly reduced FKGL score (-0.66 points).

6.3 Data Augmentation

We explored two strategies of data augmentation: enriching training data with (i) back-translated data (WikiBT), (ii) ensemble-generated data (WikiEns). Augmenting the training data using WikiEns leads to a bigger boost compared to just adding WikiBT (+1.2 vs +0.4). We also experimented with adding both synthetic datasets (WikiEnd + WikiBT) to the WikiAll training data, but the performance was worse compared to using only WikiEns.

6.4 Inference Tweaks

Finally, we describe the effect of tuning the inference hyper-parameters for our model obtained so far. Using these tweaks (Section 4.6) is one of the most crucial components of our system. Overall, it is not just able to affect the sequence generation, but also gives us the biggest boost (+2.0 points). Our final model with inference tweaks comfortably outperforms its predecessor on all datasets, demonstrating their effectiveness on the task.

7 Conclusion

This paper introduces TST, a novel approach to text simplification, by reformulating the task into a much simpler one of sequence tagging. We build TST by adapting the GECToR framework for GEC. We show that most of its performance gains are owed to simplistic data augmentations and tweaks in training and inference. These modifications allow us to derive maximal benefit from the already existing pre-trained Transformer-based encoders on large amounts of human-annotated and synthetic data, making TST a simple, powerful, easily reusable method for monolingual editing tasks. Since TST is able to progressively make simplifying edits via explicit edit-tag operations, the transformations resulting from TST are better explainable and interpretable than any other NMT-based Seq2Seq approaches. Finally, TST is fully non-autoregressive, enabling it to perform faster inference than any other state-of-the-art text simplification methods. Our empirical results demonstrate that it achieves near state-of-the-art performance on benchmark test datasets for text simplification.

A major motivation in this work was to minimize changes to the original model to keep the system simple, fast, and reproducible. Hence, we restricted our system to only use WikiAll data and its derivatives (vs. any external data like the state-of-the-art system by Martin et al. (2020b)). While we did not fully beat the state-of-the-art on the task, we believe that using larger models (eg. RoBERTa$_{\text{LARGE}}$), ensembles, or external data will likely lead to better SARI scores at the cost of speed and system complexity: ideas we plan to explore in future work.

References

Sandra M. Aluísio, Lucia Specia, T. A. S. Pardo, E. Maziero, Helena de Medeiros Caseli, and R. P. M. Fortes. 2008. A corpus analysis of simple account texts and the proposal of simplification strategies: first steps towards text simplification systems. In *SIGDOC '08*.

Sandra M. Aluísio, Lucia Specia, Thiago A. S. Pardo, Erick G. Maziero, Helena M. Caseli, and Renata P. M. Fortes. 2008. A corpus analysis of simple account texts and the proposal of simplification strategies: First steps towards text simplification systems. In *Proceedings of the 26th Annual ACM International Conference on Design of Communication*, SIGDOC '08, page 15–22, New York, NY, USA. Association for Computing Machinery.

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. *arXiv preprint arXiv:1908.04567*.

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. 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 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Beata Beigman Klebanov, Kevin Knight, and Daniel Marcu. 2004. Text simplification for information-seeking applications. In *On the Move to Meaningful Internet Systems 2004: CoopIS, DOA, and ODBASE*, pages 735–747, Berlin, Heidelberg. Springer Berlin Heidelberg.

J. D. Belder and Marie-Francine Moens. 2010. Text simplification for children. In *SIGIR 2010*.

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.

John Carroll, Guido Minnen, Darren Pearce, Yvonne Canning, Siobhan Devlin, and John Tait. 1999. Simplifying text for language-impaired readers. In *Ninth Conference of the European Chapter of the Association for Computational Linguistics*, Bergen, Norway. Association for Computational Linguistics.

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*.

Kyunghyun Cho, Bart van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder–decoder for statistical machine translation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1724–1734, Doha, Qatar. Association for Computational Linguistics.

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.

Siobhan Devlin and Gary Unthank. 2006. Helping aphasic people process online information. In *Proceedings of the 8th International ACM SIGACCESS Conference on Computers and Accessibility*, Assets '06, page 225–226, New York, NY, USA. Association for Computing Machinery.

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.

Markus Freitag, Yaser Al-Onaizan, and B. Sankaran. 2017. Ensemble distillation for neural machine translation. *ArXiv*, abs/1702.01802.

Caroline Gasperin, Erick Galani Maziero, Lucia Specia, Thiago A. S. Pardo, and Sandra M. Aluísio. 2009. Natural language processing for social inclusion : a text simplification architecture for different literacy levels.

Laurie Gerber and Eduard Hovy. 1998. Improving translation quality by manipulating sentence length. In *Machine Translation and the Information Soup*, pages 448–460, Berlin, Heidelberg. Springer Berlin Heidelberg.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: Do we need simplified corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 63–68, Beijing, China. Association for Computational Linguistics.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In *Advances in Neural Information Processing Systems 32*, pages 11181–11191. Curran Associates, Inc.

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.

Eva Hasler, Adrià de Gispert, Felix Stahlberg, Aurelien Waite, and Bill Byrne. 2017. Source sentence simplification for statistical machine translation. *Computer Speech & Language*, 45:221 – 235.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a lexical simplifier using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463, Baltimore, Maryland. Association for Computational Linguistics.

Nobuhiro Kaji, Daisuke Kawahara, Sadao Kurohashi, and Satoshi Sato. 2002. Verb paraphrase based on case frame alignment. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 215–222, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Tomoyuki Kajiwara, Hiroshi Matsumoto, and Kazuhide Yamamoto. 2013. Selecting proper lexical paraphrase for children. In *Proceedings of the 25th Conference on Computational Linguistics and Speech Processing (ROCLING 2013)*, pages 59–73, Kaohsiung, Taiwan. The Association for Computational Linguistics and Chinese Language Processing (ACLCLP).

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1537–1546, Sofia, Bulgaria. Association for Computational Linguistics.

J Peter Kincaid, Robert P Fishburne Jr, Richard L Rogers, and Brad S Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

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, Minneapolis, Minnesota. 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.

Tianyu Li, Yun Li, Jipeng Qiang, and Yun-Hao Yuan. 2018. Text simplification with self-attention-based pointer-generator networks. In *Neural Information Processing*, pages 537–545, Cham. Springer International Publishing.

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

Jonathan Mallinson, Aliaksei Severyn, Eric Malmi, and Guillermo Garrido. 2020. Felix: Flexible text editing through tagging and insertion. *arXiv preprint arXiv:2003.10687*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. 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 5054–5065, Hong Kong, China. 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. Multilingual unsupervised sentence simplification. *arXiv preprint arXiv:2005.00352*.

Aurélien Max. 2006. Writing for language-impaired readers. In *Computational Linguistics and Intelligent Text Processing*, pages 567–570, Berlin, Heidelberg. Springer Berlin Heidelberg.

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.

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.

Fernando Nogueira. 2014. Bayesian Optimization: Open source constrained global optimization tool for Python.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA â†’ Online. Association for Computational Linguistics.

Gustavo Paetzold. 2015. Reliable lexical simplification for non-native speakers. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 9–16, Denver, Colorado. Association for Computational Linguistics.

Gustavo Paetzold and Lucia Specia. 2016a. Understanding the lexical simplification needs of non-native speakers of English. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 717–727, Osaka, Japan. The COLING 2016 Organizing Committee.

Gustavo H. Paetzold and Lucia Specia. 2016b. Unsupervised lexical simplification for non-native speakers. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*, AAAI'16, page 3761–3767. AAAI Press.

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.

David Pellow and Maxine Eskenazi. 2014. An open corpus of everyday documents for simplification tasks. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)*, pages 84–93, Gothenburg, Sweden. Association for Computational Linguistics.

Sarah E. Petersen and Mari Ostendorf. 2007. Text simplification for language learners: A corpus analysis. In *In Proceedings of Workshop on Speech and Language Technology for Education*.

Jipeng Qiang. 2018. Improving neural text simplification model with simplified corpora. *arXiv preprint arXiv:1810.04428*.

Luz Rello, Ricardo Baeza-Yates, Stefan Bott, and Horacio Saggion. 2013a. Simplify or help? text simplification strategies for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, W4A '13, New York, NY, USA. Association for Computing Machinery.

Luz Rello, Ricardo Baeza-Yates, and Horacio Saggion. 2013b. The impact of lexical simplification by verbal paraphrases for people with and without dyslexia. In *Computational Linguistics and Intelligent Text Processing*, pages 501–512, Berlin, Heidelberg. Springer Berlin Heidelberg.

Joana Ribeiro, Shashi Narayan, Shay B. Cohen, and Xavier Carreras. 2018. Local string transduction as sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1360–1371, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Advaith Siddharthan. 2006. Syntactic Simplification and Text Cohesion. *Research on Language and Computation*, 4(1):77–109.

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

Sara Botelho Silveira and António Branco. 2012. Enhancing multi-document summaries with sentence simplification. In *In ICAI 2012: International Conference on Artificial Intelligence, Las Vegas*.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *Proceedings of the 9th International Conference on Computational Processing of the Portuguese Language*, PROPOR'10, page 30–39, Berlin, Heidelberg. Springer-Verlag.

Sanja Štajner, Hannah Béchara, and Horacio Saggion. 2015. A deeper exploration of the standard PB-SMT approach to text simplification and its evaluation. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 823–828, Beijing, China. Association for Computational Linguistics.

Sanja Štajner and Sergiu Nisioi. 2018. A detailed evaluation of neural sequence-to-sequence models for in-domain and cross-domain text simplification. In *Proceedings of the Eleventh International Conference on Language Resources and Evaluation (LREC-2018)*, Miyazaki, Japan. European Languages Resources Association (ELRA).

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. BLEU is not suitable for the evaluation of text simplification. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 738–744, Brussels, Belgium. 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.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In Z. Ghahramani, M. Welling, C. Cortes, N. D. Lawrence, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 27*, pages 3104–3112. Curran Associates, Inc.

Jörg Tiedemann and Santhosh Thottingal. 2020. OPUS-MT — Building open translation services for the World. In *Proceedings of the 22nd Annual Conferenec of the European Association for Machine Translation (EAMT)*, Lisbon, Portugal.

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.

Tu Vu, Baotian Hu, Tsendsuren Munkhdalai, and Hong Yu. 2018. Sentence simplification with memory-augmented neural networks. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 79–85, New Orleans, Louisiana. Association for Computational Linguistics.

Tong Wang, Ping Chen, John Rochford, and Jipeng Qiang. 2016. Text simplification using neural machine translation.

Willian Massami Watanabe, Arnaldo Candido Junior, Vinícius Rodriguez Uzêda, Renata Pontin de Mattos Fortes, Thiago Alexandre Salgueiro Pardo, and Sandra Maria Aluísio. 2009. Facilita: Reading assistance for low-literacy readers. In *Proceedings of the 27th ACM International Conference on Design of Communication*, SIGDOC '09, page 29–36, New York, NY, USA. Association for Computing Machinery.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi 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 M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing. *ArXiv*, abs/1910.03771.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 409–420, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Kristian Woodsend and Mirella Lapata. 2017. Text rewriting improves semantic role labeling. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, IJCAI'17, page 5095–5099. AAAI Press.

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. 2016a. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

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

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.

Yaoyuan Zhang, Zhenxu Ye, Yansong Feng, Dongyan Zhao, and Rui Yan. 2017. A constrained sequence-to-sequence neural model for sentence simplification. *ArXiv*, abs/1704.02312.

Sanqiang Zhao, Rui Meng, Daqing He, Saptono Andi, and Parmanto Bambang. 2018a. Integrating transformer and paraphrase rules for sentence simplification. *arXiv preprint arXiv:1810.11193*.

Sanqiang Zhao, Rui Meng, Daqing He, Andi Saptono, and Bambang Parmanto. 2018b. 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. 2020a. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *The Thirty-Fourth AAAI Conference on Artificial Intelligence, AAAI 2020, The Thirty-Second Innovative Applications of Artificial Intelligence Conference, IAAI 2020, The Tenth AAAI Symposium on Educational Advances in Artificial Intelligence, EAAI 2020, New York, NY, USA, February 7-12, 2020*, pages 9668–9675. AAAI Press.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020b. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *AAAI*, pages 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.

A Model Configurations

Table 5 describes the list of hyper-parameters used for TST-FINAL model. In Table 6, we list the inference tweaks hyper-parameters found by Bayesian Search on TurkCorpus and ASSET datasets.

Hyperparameter name	Value
learning_rate	1e-5
transformer_model	roberta-base
accumulation_size	2
batch_size	128
cold_steps_count	2
n_epoch	50
patience	3
vocab_size	5000
max_len	50
min_len	3
pieces_per_token	5
filter_brackets	0/1
seed	1/2/3/11
normalize	1

Table 5: The list of hyper-parameters used during training

B Simplification Examples

Various examples from our system are shown in Table 7. Examining the simplifications, we see reduced sentence length, sentence splitting of a complex sentence into multiple shorter sentences, and the use of simpler vocabulary. Manual comparison between TST-BASE and TST-FINAL shows that the first system tends to delete some complex words from the text. For example, "theoretically possible" gets shortened to just "possible," and "administrative district" to "district". TST-FINAL model tends to be more creative and changes phrases to simpler versions like "is theoretically possible" to "might be" or "an administrative district" to "a part of." However, this aggressive and creative strategy sometimes also generates ungrammatical output like in the last example in Table 7. While it rarely happens, but the model might also change the meaning of the original sentence. For example, replacing "the five" to "the three." It is worth noticing that the same problem of meaning change is present in the reference sentences as well: where "the five" got replaced with "one of four".

Model description	Tuned dataset	Seed	Del conf	Keep conf	Iterations	Min error probability
TST w/o InfTweaks	-	-	0	0	5	0
TST with InfTweaks	Turk	1	-0.84	-0.66	2	0.04
TST with InfTweaks	Turk	2	-0.93	-0.51	3	0.02
TST with InfTweaks	Turk	3	-0.86	-0.68	2	0.03
TST with InfTweaks	Turk	11	-0.88	-0.61	2	0.03
TST with InfTweaks	ASSET	1	-0.66	-0.9	3	0.02
TST with InfTweaks	ASSET	2	-0.72	-0.88	3	0.02
TST with InfTweaks	ASSET	3	-0.52	-0.89	3	0.04
TST with InfTweaks	ASSET	11	-0.72	-0.91	3	0.02

Table 6: Inference hyper-parameters found by the Bayesian Search on TurkCorpus and ASSET development sets

Original	he also **completed two collections** of short stories **entitled** the ribbajack & other curious yarns and seven strange and ghostly tales .
Reference	he also **wrote two books** of short stories **called ,** the ribbajack & other curious yarns and seven strange and ghostly tales .
TST-BASE	he also **wrote** two collections of short stories **called** the ribbajack & other curious yarns and seven strange and ghostly tales .
TST-FINAL	he also **wrote a series** of short stories **called** the ribbajack & other curious yarns and seven strange and ghostly tales .
Original	it **is theoretically possible** that the other editors who may have **reported** you , and the administrator who blocked you , are part of a conspiracy against someone half a world away they 've never met in person .
Reference	it is theoretically possible that the other editors who may **have written about** you, and the **officer** who blocked you, are part of **a bad plan** against someone **miles** away, they 've never met **face to face** .
TST-BASE	it is possible that the other editors who may have reported you , and the administrator who blocked you , are part of a conspiracy against someone half a world away they ' ve never met in person .
TST-FINAL	it **might be** that the other editors who may have **sent** you , and the administrator who blocked you , are part of a conspiracy against someone half a world away where they 've never met in person .
Original	as a result , although many mosques will not enforce violations , both men and women when attending a mosque must **adhere to these guidelines** .
Reference	as a result , both men and women must follow this rule when they attend a mosque , even though many mosques do not enforce these rules
TST-BASE	both men and women when **going** a mosque must **follow these rules** .
TST-FINAL	both men **,** and women **that attend** mosque **,** must **follow the law** .
Original	hinterrhein is **an administrative district in** the canton of graubünden , switzerland .
Reference	hinterrhein is a district **of** the canton of graubünden , switzerland .
TST-BASE	hinterrhein is a district **of** the canton of graubünden , switzerland .
TST-FINAL	hinterrhein is **a part of** the canton of graubünden in the switzerland .
Original	a majority of south indians speak one of the five dravidian languages — kannada , malayalam , tamil , telugu and tulu .
Reference	**many** of the south indians **are dravidians and they speak one of four** dravidian languages — kannada , malayalam , tamil **or** telugu .
TST-BASE	**most** of south indians speak one of the five dravidian languages — kannada , malayalam , tamil , telugu and tulu .
TST-FINAL	**most of the people** speak **speakers from the three** dravidian languages **spoken are ,** kannada , malayalam , tamil , telugu **,** and tulu .

Table 7: Examples of simplifications by TST