Unsupervised Neural Text Simplification

Sai Surya[†] **Abhijit Mishra**[‡] **Anirban Laha**[‡] **Parag Jain**[‡] **Karthik Sankaranarayanan**[‡]
[†]IIT Kharagpur, India [‡]IBM Research
subramanyamdvss@gmail.com
{abhijimi,anirlaha,pajain34,kartsank}@in.ibm.com

Abstract

The paper presents a first attempt towards unsupervised neural text simplification that relies only on unlabeled text corpora. The core framework is composed of a shared encoder and a pair of attentional-decoders, crucially assisted by *discrimination-based losses* and *denoising*. The framework is trained using unlabeled text collected from *en-Wikipedia* dump. Our analysis (both quantitative and qualitative involving human evaluators) on public test data shows that the proposed model can perform text-simplification at both lexical and syntactic levels, competitive to existing supervised methods. It also outperforms viable unsupervised baselines. Adding a few labeled pairs helps improve the performance further.

1 Introduction

Text Simplification (TS) deals with transforming the original text into simplified variants to increase its readability and understandability. TS is an important task in computational linguistics, and has numerous use-cases in fields of education technology, targeted content creation, language learning, where producing variants of the text with varying degree of simplicity is desired. TS systems are typically designed to simplify from two different linguistic aspects: (a) **Lexical** aspect, by replacing complex words in the input with simpler synonyms (Devlin, 1998; Candido Jr et al., 2009; Yatskar et al., 2010; Biran et al., 2011; Glavaš and Štajner, 2015), and (b) **Syntactic** aspect, by altering the inherent hierarchical structure of the sentences (Chandrasekar and Srinivas, 1997; Canning and Tait, 1999; Siddharthan, 2006; Filippova and Strube, 2008; Brouwers et al., 2014). From the perspective of sentence construction, sentence simplification can be thought to be a form of *text-transformation* that involves three major types of operations such as (a) *splitting* (Siddharthan, 2006; Petersen and Ostendorf, 2007; Narayan and Gardent, 2014) (b) *deletion/compression* (Knight and Marcu, 2002; Clarke and Lapata, 2006; Filippova and Strube, 2008; Rush et al., 2015; Filippova et al., 2015), and (c) *paraphrasing* (Specia, 2010; Coster and Kauchak, 2011; Wubben et al., 2012; Wang et al., 2016; Nisioi et al., 2017).

Most of the current TS systems require large-scale parallel corpora for training (except for systems like Glavaš and Štajner (2015) that performs only lexical-simplification), which is a major impediment in scaling to newer languages, use-cases, domains and output styles for which such large-scale parallel data do not exist. In fact, one of the popular corpus for TS in English language, *i.e.,* the Wikipedia-SimpleWikipedia aligned dataset has been prone to noise (mis-aligned instances) and inadequacy (i.e., instances having non-simplified targets) (Xu et al., 2015; Štajner et al., 2015), leading to noisy supervised models (Wubben et al., 2012). While creation of better datasets (such as, *Newsela* by Xu et al. (2015)) can always help, we explore the unsupervised learning paradigm which can potentially work with unlabeled datasets that are cheaper and easier to obtain.

At the heart of the TS problem is the need for preservation of language semantics with the goal of improving readability. From a neural-learning perspective, this entails a specially designed *auto-encoder*, which not only is capable of reconstructing the original input but also can additionally introduce variations so that the auto-encoded output is a simplified version of the input. Intuitively, both of these can be learned by looking at the structure and language patterns of a large amount of non-aligned complex and simple sentences (which are much cheaper to obtain compared to aligned parallel data). These motivations form the basis of our work.

Our approach relies only on two unlabeled text corpora - one representing relatively simpler sentences than the other (which we call complex).

The crux of the (***unsupervised***) auto-encoding framework is a shared encoder and a pair of attention-based decoders (one for each type of corpus). The encoder attempts to produce semantics-preserving representations which can be acted upon by the respective decoders (simple or complex) to generate the appropriate text output they are designed for. The framework is crucially supported by two kinds of losses: (1) ***adversarial loss*** - to distinguish between the real or fake attention context vectors for the simple decoder, and (2) ***diversification loss*** - to distinguish between attention context vectors of the simple decoder and the complex decoder. The first loss ensures that only the aspects of semantics that are necessary for simplification are passed to the simple decoder in the form of the attention context vectors. The second loss, on the other hand, facilitates passing different semantic aspects to the different decoders through their respective context vectors. Also we employ ***denoising*** in the auto-encoding setup for enabling syntactic transformations.

The framework is trained using unlabeled text collected from Wikipedia (complex) and Simple Wikipedia (simple). It attempts to perform simplification both lexically and syntactically unlike prevalent systems which mostly target them separately. We demonstrate the competitiveness of our unsupervised framework alongside supervised skylines through both automatic evaluation metrics and human evaluation studies. We also outperform another unsupervised baseline (Artetxe et al., 2018b), first proposed for neural machine translation. Further, we demonstrate that by leveraging a small amount of labeled parallel data, performance can be improved further. Our code and a new dataset containing partitioned unlabeled sets of simple and complex sentences is publicly available[1].

2 Related Work

Text Simplification has often been discussed from psychological and linguistic standpoints (L'Allier, 1980; McNamara et al., 1996; Linderholm et al., 2000). A heuristic-based system was first introduced by Chandrasekar and Srinivas (1997) which induces rules for simplification automatically extracted from annotated corpora. Canning and Tait (1999) proposed a modular system that uses NLP tools such as morphological analyzer, POS tagger plus heuristics to simplify the text both lexically and syntactically. Most of these systems (Siddharthan, 2014) are separately targeted towards lexical and syntactic simplification and are limited to splitting and/or truncating sentences. For paraphrasing based simplification, data-driven approaches were proposed like phrase-based SMT (Specia, 2010; Štajner et al., 2015) or their variants (Coster and Kauchak, 2011; Xu et al., 2016), that combine heuristic and optimization strategies for better TS. Recently proposed TS systems are based on neural *seq2seq* architecture (Bahdanau et al., 2014) which is modified for TS specific operations (Wang et al., 2016; Nisioi et al., 2017). While these systems produce state of the art results on the popular Wikipedia dataset (Coster and Kauchak, 2011), they may not be generalizable because of the noise and bias in the dataset (Xu et al., 2015) and overfitting. Towards this, Štajner and Nisioi (2018) showed that improved datasets and minor model changes (such as using reduced vocabulary and enabling copy mechanism) help obtain reasonable performance for both in-domain and cross-domain TS.

In the unsupervised paradigm, Paetzold and Specia (2016) proposed an unsupervised lexical simplification technique that replaces complex words in the input with simpler synonyms, which are extracted and disambiguated using word embeddings. However, this work, unlike ours only addresses lexical simplification and cannot be trivially extended for other forms of simplification such as splitting and rephrasing. Other works related to style transfer (Zhang et al., 2018; Shen et al., 2017; Xu et al., 2018) typically look into the problem of sentiment transformation and are not motivated by the linguistic aspects of TS, and hence not comparable to our work. As far as we know, ours is a first of its kind end-to-end solution for unsupervised TS. At this point, though supervised solutions perform better than unsupervised ones, we believe unsupervised techniques should be further explored since they hold greater potential with regards to scalability to various tasks.

3 Model Description

Our system is built based on the encode-attend-decode style architecture (Bahdanau et al., 2014) with both algorithmic and architectural changes applied to the standard model. An input sequence of word embeddings $X = \{x_1, x_2, \ldots, x_n\}$ (ob-

[1] https://github.com/subramanyamdvss/UnsupNTS

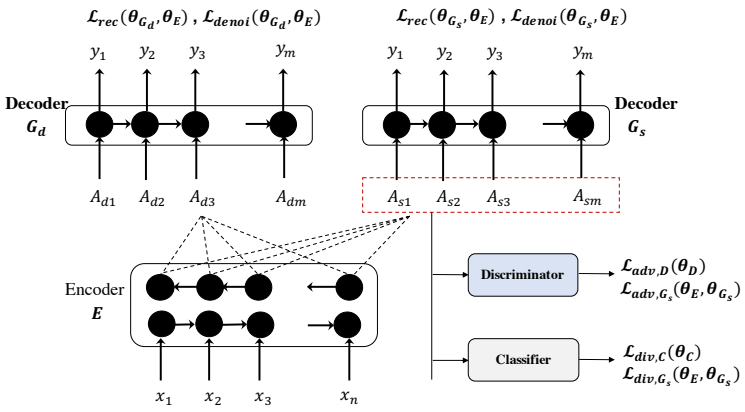

Figure 1: System Architecture. Input sentences of any domain is encoded by $\boldsymbol{E}$, and decoded by $\boldsymbol{G_s}$, $\boldsymbol{G_d}$. Discriminator $\boldsymbol{D}$ and classifier $\boldsymbol{C}$ tune the attention vectors for simplification. $\mathcal{L}$ represents loss functions. The figure only reveals one layer in $\boldsymbol{E}$, $\boldsymbol{G_s}$ and $\boldsymbol{G_d}$ for simplicity. However, the model uses two layers of GRUs (Section 3).

tained after a standard look up operation on the embedding matrix), is passed through a shared encoder ($\boldsymbol{E}$), the output representation from which is fed to two decoders ($\boldsymbol{G_s}$, $\boldsymbol{G_d}$) with attention mechanism. $\boldsymbol{G_s}$ is meant to generate a simple sentence from the encoded representation, whereas $\boldsymbol{G_d}$ generates a complex sentence. A discriminator ($\boldsymbol{D}$) and a classifier ($\boldsymbol{C}$) are also employed adversarially to distinguish between the attention context vectors computed with respect to the two decoders. Figure 1 is illustrates our system. We describe the components below.

3.1 Encode-Attend-Decode Model

Encoder $\boldsymbol{E}$ uses two layers of bi-directional GRUs (Cho et al., 2014b), and decoders $\boldsymbol{G_s}$, $\boldsymbol{G_d}$ have two layers of GRUs each. $\boldsymbol{E}$ extracts the hidden representations from an input sentence. The decoders output sentences, sequentially one word at a time. Each decoder-step involves using *global attention* to create a context-vector (hidden representations weighted by attention weights) as an input for the next decoder-step. The attention mechanism enables the decoders to focus on different parts of the input sentence. For the input sentence X with n words, the encoder produces n hidden representations, $H = \{h_1, h_2, \ldots, h_n\}$. The context vector extracted from X by a decoder $\boldsymbol{G}$ for time-step t is represented as,

$$A_t(X) = \sum_{i=1}^{n} a_{it} h_i \qquad (1)$$

where, a_{it} denotes attention weight for the hidden representation at the i^{th} input position with respect to decoder-step t. As there are two decoders, $A_{st}(X)$ and $A_{dt}(X)$ denote the context vectors computed from decoders $\boldsymbol{G_s}$ and $\boldsymbol{G_d}$ respectively for time-steps $t \in \{1 \ldots m\}$, m denoting the total number of decoding steps performed[2]. The matrices $A_s(X)$ and $A_d(X)$ represent the sequence of respective context vectors from all time-steps.

3.2 Discriminator and Classifier

A discriminator $\boldsymbol{D}$ is employed to influence the way the decoder $\boldsymbol{G_s}$ will attend to the hidden representations, which has to be different for different types of inputs to the shared encoder $\boldsymbol{E}$ (simple vs complex). The input to $\boldsymbol{D}$ is the context vector sequence matrix A_s pertaining to $\boldsymbol{G_s}$, and it produces a binary output, $\{1, 0\}$, 1 indicating the fact that the context vector sequence is close to a typical context vector sequence extracted from simple sentences seen in the dataset. $\boldsymbol{G_s}$ and $\boldsymbol{D}$ are indulged in an adversarial interplay through an adversarial loss function (see Section 4.2), analogous to GANs (Goodfellow et al., 2014), where the generator and discriminators, converge to a point where the distribution of the generations eventually resembles the distribution of the genuine samples. In our case, adversarial loss tunes the context vector sequence from a complex sentence by $\boldsymbol{G_s}$ to ultimately resemble the context vector sequence of simple sentences in the corpora. *This ensures that the resultant context vector for $\boldsymbol{G_s}$ captures only the necessary language signals to decode a simple sentence.*

A classifier ($\boldsymbol{C}$) is introduced for diversification to ensure that the way decoder $\boldsymbol{G_s}$ attends to the hidden representations remains different from

[2]For a particular X, m can differ for the two decoders.

$\boldsymbol{G_d}$. It helps distinguish between simple and complex context vector sequences with respect to $\boldsymbol{G_s}$ and $\boldsymbol{G_d}$ respectively. *The classifier diversifies the context vectors given as input to the different decoders. Intuitively, different linguistic signals are needed to decode a complex sentence vis-á-vis a simple one*. Refer Section 4.3 for more details.

Both $\boldsymbol{D}$ and $\boldsymbol{C}$ use a CNN-based classifier analogous to Kim (2014). All layers are shared between $\boldsymbol{D}$ and $\boldsymbol{C}$ except the fully-connected layer preceeding the softmax function.

3.3 Special Purpose Word-Embeddings

Pre-trained word embeddings are often seen to have positive impact on sequence-to-sequence frameworks (Cho et al., 2014a; Qi et al., 2018). However, traditional embeddings are not good at capturing relations like synonymy (Tissier et al., 2017), which are essential for simplification. For this, our word-embeddings are trained using the *Dict2Vec* framework[3]. *Dict2Vec* fine-tunes the embeddings through the help of an external lexicon containing weak and strong synonymy relations. The system is trained on our whole unlabeled datasets and with seed synonymy dictionaries provided by Tissier et al. (2017). Our encoder and decoders share the same word embeddings. Moreover, the embeddings at the input side are kept static but the decoder embeddings are updated as training progresses. Details about hyperparameters are given in Section 5.2.

4 Training Procedure

Let $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{\mathcal{D}}$ be sets of simple and complex sentences respectively from large scale unlabeled repositories of simple and complex sentences. Let X_s denote a sentence sampled from the set of simple sentences $\boldsymbol{\mathcal{S}}$ and X_d be a sentence sampled from the set of complex sentences $\boldsymbol{\mathcal{D}}$. Let $\boldsymbol{\theta_E}$ denote the parameters of $\boldsymbol{E}$ and $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ denote the parameters of $\boldsymbol{G_s}$ and $\boldsymbol{G_d}$ respectively. Also, $\boldsymbol{\theta_C}$ and $\boldsymbol{\theta_D}$ are the parameters of the discriminator and the classifier modules. Training the model involves optimization of the above parameters with respect to the following losses and *denoising*, which are explained below.

4.1 Reconstruction Loss

Reconstruction Loss is imposed on both $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$ paths. $\boldsymbol{E-G_s}$ is trained to reconstruct sentences from $\boldsymbol{\mathcal{S}}$ and $\boldsymbol{E-G_d}$ is trained to reconstruct sentences from $\boldsymbol{\mathcal{D}}$. Let $P_{\boldsymbol{E-G_s}}(X)$ and $P_{\boldsymbol{E-G_d}}(X)$ denote the reconstruction probabilities of an input sentence X estimated by the $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$ models respectively. Reconstruction loss for $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$, denoted by $\mathcal{L}_{rec}$ is computed as follows.

$$\mathcal{L}_{rec}(\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}, \boldsymbol{\theta_{G_d}}) = \mathbb{E}_{X_s \sim \boldsymbol{\mathcal{S}}}[\log P_{\boldsymbol{E-G_s}}(X_s)] + \mathbb{E}_{X_d \sim \boldsymbol{\mathcal{D}}}[\log P_{\boldsymbol{E-G_d}}(X_d)] \quad (2)$$

4.2 Adversarial Loss

Adversarial Loss is imposed upon the context vectors for $\boldsymbol{G_s}$. The idea is that, context vectors extracted even for a complex input sentence by $\boldsymbol{G_s}$ should resemble the context vectors from a simple input sentence. The discriminator $\boldsymbol{D}$ is trained to distinguish the *fake* (complex) context vectors from the *real* (simple) context vectors. $\boldsymbol{E-G_s}$ is trained to perplex the discriminator $\boldsymbol{D}$, and eventually, at convergence, learns to produce real-like (simple) context vectors from complex input sentences. In practice, we observe that adversarial loss indeed assists $\boldsymbol{E-G_s}$ in simplification by encouraging sentence shortening. Let $A_s(.)$ be a sequence of context vectors as defined in Section 3.1. Adversarial losses for $\boldsymbol{E-G_s}$, denoted by $\mathcal{L}_{adv,G_s}$ and for discriminator $\boldsymbol{D}$, denoted by $\mathcal{L}_{adv,D}$ are as follows.

$$\mathcal{L}_{adv,D}(\boldsymbol{\theta_D}) = \mathbb{E}_{X_s \sim \boldsymbol{\mathcal{S}}}[\log(\boldsymbol{D}(A_s(X_s)))] + \mathbb{E}_{X_d \sim \boldsymbol{\mathcal{D}}}[\log(1 - \boldsymbol{D}(A_s(X_d))] \quad (3)$$

$$\mathcal{L}_{adv,G_s}(\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}) = \mathbb{E}_{X_d \sim \boldsymbol{\mathcal{D}}}[\log(\boldsymbol{D}(A_s(X_d)))] \quad (4)$$

4.3 Diversification Loss

Diversification Loss is imposed by the classifier $\boldsymbol{C}$ on context vectors extracted by $\boldsymbol{G_d}$ from complex input sentences in contrast with context vectors extracted by $\boldsymbol{G_s}$ from simple input sentences. This helps $\boldsymbol{E-G_s}$ to learn to generate simple context vectors distinguishable from complex context vectors. Let $A_s(.)$ and $A_d(.)$ be sequence of context vectors as defined in Section 3.1. Losses for classifier C, denoted by $\mathcal{L}_{div,C}$ and for model $\boldsymbol{E-G_s}$ denoted by $\mathcal{L}_{div,G_s}$ are computed as follows.

$$\mathcal{L}_{div,C}(\boldsymbol{\theta_C}) = \mathbb{E}_{X_s \sim \boldsymbol{\mathcal{S}}}[\log(\boldsymbol{C}(A_s(X_s)))] + \mathbb{E}_{X_d \sim \boldsymbol{\mathcal{D}}}[\log(1 - \boldsymbol{C}(A_d(X_d)))] \quad (5)$$

$$\mathcal{L}_{div,G_s}(\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}) = \mathbb{E}_{X_d \sim \boldsymbol{\mathcal{D}}}[\log(\boldsymbol{C}(A_d(X_d)))] \quad (6)$$

[3] https://github.com/tca19/dict2vec

Algorithm 1 Unsupervised simplification algorithm using denoising, reconstruction, adversarial and diversification losses.

Input: simple dataset $\boldsymbol{\mathcal{S}}$, complex dataset $\boldsymbol{\mathcal{D}}$.

Initialization phase:

repeat

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{denoi}$

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{rec}$

Update $\boldsymbol{\theta_D}$, $\boldsymbol{\theta_C}$ using $\mathcal{L}_{adv,D}$ $\mathcal{L}_{div,C}$

until specified number of steps are completed

Adversarial phase:

repeat

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{denoi}$

Update $\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}, \boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{adv,G_s}$, $\mathcal{L}_{div,G_s}$, $\mathcal{L}_{rec}$

Update $\boldsymbol{\theta_D}$, $\boldsymbol{\theta_C}$ using $\mathcal{L}_{adv,D}$, $\mathcal{L}_{div,C}$

until specified number of steps are completed

4.4 Denoising

Denoising has proven to be helpful to learn syntactic / structural transformation from the source side to the target side (Artetxe et al., 2018b). Syntactic transformation often requires reordering the input, which the denoising procedure aims to capture. Denoising involves arbitrarily reordering the inputs and reconstructing the original (unperturbed) input from such reordered inputs. In our implementation, the source sentence is reordered by swapping bigrams in the input sentences. The following loss function are used in denoising. Let $P_{\boldsymbol{E-G_s}}(X|noise(X))$ and $P_{\boldsymbol{E-G_d}}(X|noise(X))$ denote the probabilities that a perturbed input X can be reconstructed by $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$ respectively. Denoising loss for models $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$, denoted by $\mathcal{L}_{denoi}(\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}, \boldsymbol{\theta_{G_d}})$ is computed as follows.

$$\mathcal{L}_{denoi} = \mathbb{E}_{X_s \sim \boldsymbol{\mathcal{S}}}[\log P_{\boldsymbol{E-G_s}}(X_s|noise(X_s))] + \mathbb{E}_{X_d \sim \boldsymbol{\mathcal{D}}}[\log P_{\boldsymbol{E-G_d}}(X_d|noise(X_d))] \quad (7)$$

Figure 1 depicts the overall architecture and the losses described above; the training procedure is described in Algorithm 1. The *initialization phase* involves training the $\boldsymbol{E-G_s}$, $\boldsymbol{E-G_d}$ using the reconstruction and denoising losses only. Next, training of $\boldsymbol{D}$ and $\boldsymbol{C}$ happens using the respective adversarial or diversification losses. These losses are not used to update the decoders at this point. This gives the discriminator, classifier and decoders time to learn independent of each other. In the *adversarial phase*, adversarial and diversification losses are introduced alongside denoising and reconstruction losses for fine-tuning the encoder and decoders. Algorithm 1 is intended to produce the following results: i) $\boldsymbol{E-G_s}$ should simplify its input (irrespective of whether it is simple or complex), and ii) $\boldsymbol{E-G_d}$ should act as an auto-encoder in complex sentence domain. The discriminator and classifier enables preserving the appropriate aspects of semantics necessary for each of these pathways through proper modulation of the attention context vectors.

A key requirement for a model like ours is that the dataset used has to be partitioned into two sets, containing relatively simple and complex sentences. The rationale behind having two decoders is that while $\boldsymbol{G_s}$ will try to introduce simplified constructs (may be at the expense of loss of semantics), $\boldsymbol{G_d}$ will help preserve the semantics. The idea behind using the discriminator and classifier is to retain signals related to language simplicity from which $\boldsymbol{G_s}$ will construct simplified sentences. Finally, denoising will help tackle nuances related to syntactic transfer from complex to simple direction. We remind the readers that, TS, unlike machine translation, needs complex syntactic operations such as sentence splitting, rephrasing and paraphrasing, which can not be tackled by the losses and denoising alone. Employing additional explicit mechanisms to handle these in the pipeline is out of the scope of this paper since we seek a *prima-facie* judgement of our architecture based on how much simplification knowledge can be gained just from the data.

4.5 Training with Minimal Supervision

Our system, by design, is highly data-driven, and like any other sequence-to-sequence learning based system, can also leverage labeled data. We propose a semi-supervised variant of our system that could gain additional knowledge of simplification through the help of a small amount of labeled data (in the order of a few thousands). The system undergoes training following steps similar to Algorithm 1, except that it adds another step of optimizing the *cross entropy loss* for both the $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$ pathways by using the reference texts available in the labeled dataset. This step is carried out in the adversarial phase along with other steps (See Algorithm 2).

The cross-entropy loss is imposed on both

$\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$ paths using parallel dataset (details mentioned in Section 5.1) denoted by $\Delta = (\boldsymbol{\mathcal{S}_p}, \boldsymbol{\mathcal{D}_p})$. For a given parallel simplification sentence pair (X_s, X_d), let $P_{\boldsymbol{E-G_s}}(X_s|X_d)$ and $P_{\boldsymbol{E-G_d}}(X_d|X_s)$ denote the probabilities that X_s is produced from X_d by the $\boldsymbol{E-G_s}$ and the reverse is produced by the $\boldsymbol{E-G_d}$ respectively. Cross-Entropy loss for $\boldsymbol{E-G_s}$ and $\boldsymbol{E-G_d}$ denoted by $\mathcal{L}_{cross}(\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}, \boldsymbol{\theta_{G_d}})$ is computed as follows:

$$\mathcal{L}_{cross} = \mathbb{E}_{(X_s,X_d)\sim\boldsymbol{\Delta}}[\log P_{\boldsymbol{E-G_s}}(X_s|X_d)] + \mathbb{E}_{(X_s,X_d)\sim\boldsymbol{\Delta}}[\log P_{\boldsymbol{E-G_d}}(X_d|X_s)] \quad (8)$$

Algorithm 2 Semi-supervised simplification algorithm using denoising, reconstruction, adversarial and diversification losses followed by cross-entropy loss using parallel data.

Input: simple dataset $\boldsymbol{\mathcal{S}}$, complex dataset $\boldsymbol{\mathcal{D}}$, parallel dataset $\Delta = (\boldsymbol{\mathcal{S}_p}, \boldsymbol{\mathcal{D}_p})$

Initialization phase:

repeat

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{denoi}$

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{rec}$

Update $\boldsymbol{\theta_D}$, $\boldsymbol{\theta_C}$ using $\mathcal{L}_{adv,D}$ $\mathcal{L}_{div,C}$

until specified number of steps are completed

Adversarial phase:

repeat

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{denoi}$

Update $\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}, \boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{adv,G_s}$, $\mathcal{L}_{div,G_s}$, $\mathcal{L}_{rec}$

Update $\boldsymbol{\theta_D}$, $\boldsymbol{\theta_C}$ using $\mathcal{L}_{adv,D}$, $\mathcal{L}_{div,C}$

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_s}}$ using $\mathcal{L}_{cross}$

Update $\boldsymbol{\theta_E}$, $\boldsymbol{\theta_{G_d}}$ using $\mathcal{L}_{cross}$

until specified number of steps are completed

5 Experiment Setup

In this section we describe the dataset, architectural choices, and model hyperparameters. The implementation of the experimental setup is publicly available[4].

5.1 Dataset

For training our system, we created an unlabeled dataset of simple and complex sentences by partitioning the standard *en-wikipedia* dump. Since partitioning requires a metric for measuring text simpleness we categorize sentences based on their readability scores. For this we use the Flesch Readability Ease (henceforth abbreviated as FE) (Flesch, 1948). Sentences with lower FE values (up to 10) are categorized as complex and sentences with FE values greater than 70 are categorized as simple[5]. The FE bounds are decided through trial and error through manual inspection of the categorized sentences. Table 1 shows dataset statistics. Even though the dataset was created with some level of human mediation, the manual effort is insignificant compared to that needed to create a parallel corpus.

Category	#Sents	Avg. Words	Avg. FE	FE-Range
Simple	720k	18.23	76.67	74.9-79.16
Complex	720k	35.03	7.26	5.66-9.93

Table 1: Statistics showing number of sentences, average words per sentence, and average FE score, FE score limits for complex and simple datasets used for training.

To train the system with minimal supervision (Section 4.5), we extract $10,000$ pairs of sentences from various datasets such as Wikipedia-SimpleWikipedia dataset introduced in Hwang et al. (2015) and the Split-Rephrase dataset by Narayan et al. (2017)[6]. The Wikipedia-SimpleWikipedia was filtered following Nisioi et al. (2017) and 4000 examples were randomly picked from the filtered set. From the Split-Rephrase dataset, examples containing one compound/complex sentence at the source side and two simple sentences at the target side were selected and 6000 examples were randomly picked from the selected set. The Split-Rephrase dataset is used to promote sentence splitting in the proposed system.

To select and evaluate our models, we use the test and development sets[7] released by (Xu et al., 2016). The test set (359 sentences) and development set (2000 sentences) have 8 simplified reference sentences for each source sentence.

5.2 Hyperparameter Settings

For all the variants, we use a hidden state of size 600 and word-embedding size of 300. Classifier C

[4]https://github.com/subramanyamdvss/UnsupNTS

[5]FE has its shortcomings to fully judge simpleness, but we nevertheless employ it in the absence of stronger metrics

[6]https://github.com/shashiongithub/Split-and-Rephrase

[7]We acknowledge that other recent datasets such as Newsela could have been used for development and evaluation. We could not get access to the dataset unfortunately.

and discriminator D use convolutional layers with filters sizes from 1 to 5. 128 filters of each size are used in the CNN-layers. Other training related hyper parameters include learning rate of 0.00012 for $\boldsymbol{\theta_E}, \boldsymbol{\theta_{G_s}}, \boldsymbol{\theta_{G_d}}$, 0.0005 for $\boldsymbol{\theta_D}, \boldsymbol{\theta_C}$ and batch size of 36. For learning the word-embedding using *Dict2Vec* training, the window size is set to 5. Our experiments used at most 13 GB of GPU memory. The Initialization phase and Adversarial phase took 6000 and 8000 steps in batches respectively for both UNTS and UNTS+10K systems.

5.3 Evaluation Metrics

For automatic evaluation of our system on the test data, we used four metrics, (a) SARI (b) BLEU (c) FE Difference (d) Word Difference, which are briefly explained below.

SARI (Xu et al., 2016) is an automatic evaluation metric designed to measure the simpleness of the generated sentences. SARI requires access to source, predictions and references for evaluation. Computing SARI involves penalizing the n-gram additions to source which are inconsistent with the references. Similarly, deletions and keep operations are penalized. The overall score is a balanced sum of all the penalties. BLEU (Papineni et al., 2002), a popular metric to evaluate generations and translations is used to measure the correctness of the generations by measuring overlaps between the generated sentences and (multiple) references.

We also compute the average FE score difference between predictions and source in our evaluations. FE-difference measures whether the changes made by the model increase the readability ease of the generated sentence. *Word Difference* is the average difference between number of words in the source sentence and generation. It is a simple and approximate metric proposed to detect if sentence shortening is occurring or not. Generations with lesser number of changes can still have high SARI and BLEU. Models with such generations can be ruled out by imposing a threshold on the word-diff metric.

Models with high word-diff, SARI and BLEU are picked during model-selection (with validation data). Model selection also involved manually examining the quality and relevance of generations.

We carry out a qualitative analysis of our system through human evaluation. For this the first 50 test samples were selected from the test data. Output of the seven systems reported in Table 2 along with the sources are presented to two native English speakers who would provide two ratings for each output: (a) *Simpleness*, a binary score [0-1] indicating whether the output is a simplified version of the input or not, (b) *Grammaticality* of the output in the range of [1-5], in the increasing order of fluency (c) *Relatedness* score in the range of [1-5] showing if the overall semantics of the input is preserved in the output or not.

5.4 Model Variants

Using our design, we propose two different variants for evaluation: (i) Unsupervised Neural TS (UNTS) with SARI as the criteria for model selection, (ii) UNTS with minimal supervision using 10000 labelled examples (UNTS+10K). Models selected using other selection criteria such as BLEU resulted in similar and/or reduced performance (details skipped for brevity).

We carried out the following basic post-processing steps on the generated outputs. The OOV(out of vocabulary) words in the generations are replaced by the source words with high attention weights. Words repeated consecutively in the generated sentences are merged.

5.5 Systems for Comparison

In the absence of any other direct baseline for end-to-end TS, we consider the following unsupervised baselines. We consider the unsupervised NMT framework proposed by (Artetxe et al., 2018b) as a baseline. It uses techniques such as *backtranslation* and *denoising* techniques to synthesize more training examples. To use this framework, we treated the set of simple and complex sentences as two different languages. Same model configuration as reported by Artetxe et al. (2018b) is used. We use the term UNMT for this system.

Similar to the UNMT system, we also consider unsupervised statistical machine translation (termed as USMT) proposed by Artetxe et al. (2018a), with default parameter setting. Another system, based on the cross alignment technique proposed by Shen et al. (2017) is also used for comparison. The system is originally proposed for the task of sentiment translation. We term this system as ST.

We also compare our approach with existing supervised and unsupervised lexical simplifications like LIGHTLS (Glavaš and Štajner, 2015), Neural Text Simplification or NTS (Nisioi et al., 2017), Syntax based Machine Translation or SBMT (Xu

System	FE-diff	SARI	BLEU	Word-diff
UNTS+10K	10.45	35.29	76.13	2.38
UNTS	11.15	33.8	74.24	**3.55**
UNMT	6.60	33.72	70.84	0.74
USMT	13.84	32.11	87.36	-0.01
ST	54.38	14.97	0.73	5.61
NTS	5.37	36.1	79.38	2.73
SBMT	**17.68**	**38.59**	73.62	-0.84
PBSMT	9.14	34.07	67.79	2.26
LIGHTLS	3.01	34.96	**83.54**	-0.02

Table 2: Comparison of evaluation metrics for proposed systems (UNTS), unsupervised baseline (UNMT,USMT, and ST) and existing supervised and the unsupervised lexical simplification system LIGHTLS.

System	Simpleness	Fluency	Relatedness
UNTS+10K	**57**%	4.13	3.93
UNTS	47%	3.86	3.73
UNMT	40%	3.8	**4.06**
NTS	49%	4.13	3.26
SBMT	53%	**4.26**	**4.06**
PBSMT	53%	3.8	3.93
LIGHTLS	6%	4.2	3.33

Table 3: Average human evaluation scores for simpleness and grammatical correctness (fluency) and semantic relatedness between the output and input.

et al., 2016), and Phrase-based SMT simplification or PBSMT (Wubben et al., 2012). All the systems are trained using the Wikipedia-SimpleWikipedia dataset (Hwang et al., 2015). The test set is same for all of these and our models.

6 Results

Table 2 shows evaluation results of our proposed approaches along with existing supervised and unsupervised alternatives. We observe that unsupervised baselines such as UNMT and USMT often, after attaining convergence, recreates sentences similar to the inputs. This explains why they achieve higher BLEU and reduced word-difference scores. The ST system did not converge for our dataset after significant number of epochs which affected the performance metrics. The system often produces short sentences which are simple but do not retain important phrases.

Other supervised systems such as SBMT and NTS achieve better content reduction as shown through SARI, BLEU and FE-diff scores; this is expected. However, it is still a good sign that the scores for the unsupervised system UNTS are not far from the supervised skylines. The higher word-diff scores for the unsupervised system also indicate that it is able to perform content reduction (a form of syntactic simplification), which is crucial to TS. This is unlike the existing unsupervised LIGHTLS system which often replaces nouns with related non-synonymous nouns; sometimes increasing the complexity and affecting the meaning. Finally, it is worth noting that aiding the system with a very small amount of labeled data can also benefit our unsupervised pipeline, as suggested by the scores for the UNTS+10K system.

In Table 3, the first column represents what percentage of output form is a simplified version of the input. The second and third columns present the average fluency (grammaticality) scores given by human evaluators and semantic relatedness with input scored through automatic means. Almost all systems are able to produce sentences that are somewhat grammatically correct and retain phrases from input. Supervised systems like PBSMT, as expected, simplify the sentences to the maximum extent. However, our unsupervised variants have scores competitive to the supervised skylines, which is a positive sign.

Table 4 shows an anecdotal example, containing outputs from the seven systems. As can be seen, the quality of output from our unsupervised variants, is far from that of the reference output. However, the attempts towards performing lexical simplification (by replacing the word "Neverthless" with "However") and simplification of multi-word phrases ("Tagore emulated numerous styles" getting translated to "Tagore replaced many styles") are quite visible and encouraging. Table 5 presents a few examples demonstrating the capabilities of our system in performing simplifications at lexical and syntactic level. We do observe that such operations are carried out only for a few instances in our test data.

To understand the impact of additional parallel data, we further carried out analysis on adding different amounts of parallel data (2K, 5K and 10 pairs). The findings are summarized in Table 6. This analysis indicates that the system can improve over time with addition of more data. From the trained models, models with similar word-diff are chosen for fair comparison. Our observation is that, with increasing data, BLEU as well as SARI increases.

System	Output
Input	Nevertheless , Tagore emulated numerous styles , including craftwork from northern New Ireland , Haida carvings from the west coast of Canada (British Columbia) , and woodcuts by Max Pechstein .
Reference	Nevertheless , Tagore copied many styles , such as crafts from northern New Ireland , Haida carvings from the west coast of Canada and wood carvings by Max Pechstein .
UNTS+10K	Nevertheless , Tagore replaced many styles , including craftwork from northern New Ireland , Haida carved from the west coast of Canada (British Columbia) .
UNTS	However , Tagore notably numerous styles , including craftwork from northern New Ireland , Haida carved from the west coast of Canada (British) .
UNMT	However , Tagore featured numerous styles including craftwork from northern New Ireland , Haida from the west coast of Canada (British Columbia) max by Max Pechstein .
USMT	Nevertheless, Mgr emulated numerous styles, including craftwork from northern New Ireland, Haida carvings from the west coast of Canada (British Columbia), and etchings by Max Pechstein.
NTS	However , Tagore wrote many styles , including craftwork from northern New Ireland , Haida carvings from the west coast of Canada (British Columbia) .
SBMT	However , Tagore emulated many styles , such as craftwork in north New Ireland , Haida prints from the west coast of Canada (British Columbia) , and woodcuts by Max Pechstein .
PBSMT	Nevertheless , he copied many styles , from new craftwork , Haida carvings from the west coast of Canada in British Columbia and woodcuts by Max Pechstein .
LIGHTLS	However , Tagore imitated numerous styles , including craftwork from northern New Ireland , Haida sculptures from the west coast of Canada (British Columbia) , and engravings by Max Pechstein .

Table 4: Example predictions from different systems

Type of Simplification	Source	Prediction
Splitting	Calvin Baker is an American novelist .	Calvin Baker is an American . American Baker is a birthplace .
Sentence Shortening	During an interview , Edward Gorey mentioned that Bawden was one of his favorite artists , **lamenting the fact that not many people remembered or knew about this fine artist .**	During an interview , Edward Gorey mentioned that Bawden was one of his favorite artists .
Lexical Replacement	In **architectural** decoration Small pieces of colored and **iridescent** shell have been used to create **mosaics** and inlays , which have been used to decorate walls , furniture and boxes .	In **impressive** decoration Small pieces of colored and **reddish** shell have been used to create **statues** and inlays , which have been used to decorate walls , furniture and boxes .

Table 5: Examples showing different types of simplifications performed by the best model UNTS+10K

System	FE-diff	SARI	BLEU	Word-diff
UNTS	11.15	33.8	74.24	3.55
UNTS+2K	11.64	34.17	72.63	3.26
UNTS+5K	11.69	34.39	70.96	3.01
UNTS+10K	11.65	35.14	75.71	3.05

Table 6: Effect of variation in labeled data considered as additional help during training the unsupervised systems

Results for ablations on adversarial and diversification loss are also mentioned in Appendix A.

7 Conclusion

In this paper, we made a novel attempt towards unsupervised text simplification. We gathered unlabeled corpora containing simple and complex sentences and used them to train our system that is based on a shared encoder and two decoders. A novel training scheme is proposed which allows the model to perform content reduction and lexical simplification simultaneously through our proposed losses and denoising. Experiments were conducted for multiple variants of our system as well as known unsupervised baselines and supervised systems. Qualitative and quantitative analysis of the outputs for a publicly available test data demonstrate that our models, though unsupervised, can perform better than or competitive to these baselines. In future, we would like to improve the system further by incorporating better architectural designs and training schemes to tackle complex simplification operations.

8 Acknowledgements

We thank researchers at IBM IRL, IIT Kharagpur, Vishal Gupta and Dr. Sudeshna Sarkar for helpful discussions in this project.

References

Mikel Artetxe, Gorka Labaka, and Eneko Agirre. 2018a. Unsupervised statistical machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3632–3642.

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2018b. Unsupervised neural machine translation. In *Proceedings of the Sixth International Conference on Learning Representations*.

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

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *ACL*, pages 496–501. Association for Computational Linguistics.

Laetitia Brouwers, Delphine Bernhard, Anne-Laure Ligozat, and Thomas François. 2014. Syntactic sentence simplification for french. In *Proceedings of the 3rd Workshop on Predicting and Improving Text Readability for Target Reader Populations (PITR)@ EACL 2014*, pages 47–56.

Arnaldo Candido Jr, Erick Maziero, Caroline Gasperin, Thiago AS Pardo, Lucia Specia, and Sandra M Aluisio. 2009. Supporting the adaptation of texts for poor literacy readers: a text simplification editor for brazilian portuguese. In *Innovative Use of NLP for Building Educational Applications*, pages 34–42. Association for Computational Linguistics.

Yvonne Canning and John Tait. 1999. Syntactic simplification of newspaper text for aphasic readers. In *Customised Information Delivery*, pages 6–11.

Raman Chandrasekar and Bangalore Srinivas. 1997. Automatic induction of rules for text simplification1. *Knowledge-Based Systems*, 10(3):183–190.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. 2014a. On the properties of neural machine translation: Encoder–decoder approaches. In *Proceedings of SSST-8, Eighth Workshop on Syntax, Semantics and Structure in Statistical Translation*, pages 103–111.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014b. 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.

James Clarke and Mirella Lapata. 2006. Models for sentence compression: A comparison across domains, training requirements and evaluation measures. In *COLING*, pages 377–384. Association for Computational Linguistics.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *ACL*, pages 665–669. Association for Computational Linguistics.

Siobhan Devlin. 1998. The use of a psycholinguistic database in the simplification of text for aphasic readers. *Linguistic databases*.

Katja Filippova, Enrique Alfonseca, Carlos A Colmenares, Lukasz Kaiser, and Oriol Vinyals. 2015. Sentence compression by deletion with lstms. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 360–368.

Katja Filippova and Michael Strube. 2008. Dependency tree based sentence compression. In *INLG*, pages 25–32. Association for Computational Linguistics.

Rudolph Flesch. 1948. A new readability yardstick. *Journal of applied psychology*, 32(3):221.

Goran Glavaš and Sanja Štajner. 2015. Simplifying lexical simplification: do we need simplified corpora? In *ACL*, volume 2, pages 63–68.

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL-HLT*, pages 211–217.

Yoon Kim. 2014. Convolutional neural networks for sentence classification. *arXiv:1408.5882*.

Kevin Knight and Daniel Marcu. 2002. Summarization beyond sentence extraction: A probabilistic approach to sentence compression. *Artificial Intelligence*, 139(1):91–107.

J. L'Allier. 1980. An evaluation study of a computer-based lesson that adjusts read- ing level by monitoring on task reader characteristics. *Ph.D. Thesis*.

Tracy Linderholm, Michelle Gaddy Everson, Paul Van Den Broek, Maureen Mischinski, Alex Crittenden, and Jay Samuels. 2000. Effects of causal text revisions on more-and less-skilled readers' comprehension of easy and difficult texts. *Cognition and Instruction*, 18(4):525–556.

Danielle S McNamara, Eileen Kintsch, Nancy Butler Songer, and Walter Kintsch. 1996. Are good texts always better? interactions of text coherence, background knowledge, and levels of understanding in learning from text. *Cognition and instruction*, 14(1):1–43.

Shashi Narayan and Claire Gardent. 2014. Hybrid simplification using deep semantics and machine translation. In *ACL*, volume 1, pages 435–445.

Shashi Narayan, Claire Gardent, Shay Cohen, and Anastasia Shimorina. 2017. Split and rephrase. In *EMNLP 2017: Conference on Empirical Methods in Natural Language Processing*, pages 617–627.

Sergiu Nisioi, Sanja Štajner, Simone Paolo Ponzetto, and Liviu P Dinu. 2017. Exploring neural text simplification models. In *ACL*, volume 2, pages 85–91.

Gustavo H Paetzold and Lucia Specia. 2016. Unsupervised lexical simplification for non-native speakers. In *AAAI*, pages 3761–3767.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *ACL*, pages 311–318. Association for Computational Linguistics.

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

Ye Qi, Devendra Sachan, Matthieu Felix, Sarguna Padmanabhan, and Graham Neubig. 2018. When and why are pre-trained word embeddings useful 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 2 (Short Papers)*, pages 529–535.

Alexander M Rush, Sumit Chopra, and Jason Weston. 2015. A neural attention model for abstractive sentence summarization. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 379–389.

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

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. *ITL-International Journal of Applied Linguistics*, 165(2):259–298.

Lucia Specia. 2010. Translating from complex to simplified sentences. In *Computational Processing of the Portuguese Language*, pages 30–39. Springer.

Sanja Štajner, Hannah Bechara, and Horacio Saggion. 2015. A deeper exploration of the standard pb-smt approach to text simplification and its evaluation. In *ACL-IJCNLP*, volume 2, pages 823–828.

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 Language Resources Association (ELRA).

Julien Tissier, Christophe Gravier, and Amaury Habrard. 2017. Dict2vec : Learning word embeddings using lexical dictionaries. In *EMNLP*, pages 254–263.

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

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: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

Jingjing Xu, SUN Xu, Qi Zeng, Xiaodong Zhang, Xuancheng Ren, Houfeng Wang, and Wenjie Li. 2018. Unpaired sentiment-to-sentiment translation: A cycled reinforcement learning approach. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 979–988.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association of Computational Linguistics*, 3(1):283–297.

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

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from wikipedia. In *NAACL-HLT*, pages 365–368. Association for Computational Linguistics.

Zhirui Zhang, Shuo Ren, Shujie Liu, Jianyong Wang, Peng Chen, Mu Li, Ming Zhou, and Enhong Chen. 2018. Style transfer as unsupervised machine translation. *arXiv preprint arXiv:1808.07894*.

A Ablation Studies

The following table shows results of the proposed system with ablations on adversarial loss (UNTS-ADV) and diversification loss (UNTS-DIV).

System	FE-diff	SARI	BLEU	Word-diff
UNTS+10K	10.45	35.29	76.13	2.38
UNTS-DIV+10K	11.32	35.24	75.59	2.61
UNTS-ADV+10K	10.32	35.08	76.19	2.64
UNTS	11.15	33.8	74.24	3.55
UNTS-DIV	14.15	34.38	68.65	3.46
UNTS-ADV	12.13	34.74	73.21	2.72

Table 7: UNTS-ADV does not use the adversarial loss, UNTS-DIV does not use the diversification loss.