EditNTS: An Neural Programmer-Interpreter Model for Sentence Simplification through Explicit Editing

Yue Dong[1], Zichao Li[2], Mehdi Rezagholizadeh[2], Jackie Chi Kit Cheung[1]

[1]MILA, McGill

`yue.dong2@mail.mcgill.ca, jcheung@cs.mcgill.ca`

[2]Huawei Noah's Ark Lab

`{li.zichao, mehdi.rezagholizadeh}@huawei.com`

Abstract

We present the first sentence simplification model that learns explicit edit operations (`ADD`, `DELETE`, and `KEEP`) via a neural programmer-interpreter approach. Most current neural sentence simplification systems are variants of sequence-to-sequence models adopted from machine translation. These methods learn to simplify sentences as a byproduct of the fact that they are trained on complex-simple sentence pairs. By contrast, our neural programmer-interpreter is directly trained to predict explicit edit operations on targeted parts of the input sentence, resembling the way that humans might perform simplification and revision. Our model outperforms previous state-of-the-art neural sentence simplification models (without external knowledge) by large margins on three benchmark text simplification corpora in terms of SARI (+0.95 WikiLarge, +1.89 WikiSmall, +1.41 Newsela), and is judged by humans to produce overall better and simpler output sentences[1].

1 Introduction

Sentence simplification aims to reduce the reading complexity of a sentence while preserving its meaning. Simplification systems can benefit populations with limited literacy skills (Watanabe et al., 2009), such as children, second language speakers and individuals with language impairments including dyslexia (Rello et al., 2013), aphasia (Carroll et al., 1999) and autism (Evans et al., 2014).

Inspired by the success of machine translation, many text simplification (TS) systems treat sentence simplification as a monolingual translation task, in which complex-simple sentence pairs

are presented to the models as source-target pairs (Zhang and Lapata, 2017). Two major machine translation (MT) approaches are adapted into TS systems, each with its advantages: statistical machine translation (SMT)-based models (Zhu et al., 2010; Wubben et al., 2012; Narayan and Gardent, 2014; Xu et al., 2016) can easily integrate human-curated features into the model, while neural machine translation (NMT)-based models (Nisioi et al., 2017; Zhang and Lapata, 2017; Vu et al., 2018) can operate in an end-to-end fashion by extracting features automatically. Nevertheless, MT-based models must learn the simplifying operations that are embedded in the parallel complex-simple sentences *implicitly*. These operations are relatively infrequent, as a large part of the original complex sentence usually remains unchanged in the simplification process (Zhang et al., 2017). This leads to MT-based models that often produce outputs that are identical to the inputs (Zhao et al., 2018), which is also confirmed in our experiments.

We instead propose a novel end-to-end Neural Programmer-Interpreter (Reed and de Freitas, 2016) that learns to *explicitly* generate edit operations in a sequential fashion, resembling the way that a human editor might perform simplifications on sentences. Our proposed framework consists of a programmer and an interpreter that operate alternately at each time step: the programmer predicts a simplifying edit operation (program) such as `ADD`, `DELETE`, or `KEEP`; the interpreter executes the edit operation while maintaining a context and an edit pointer to assist the programmer for further decisions. Table 1 shows sample runs of our model.

Intuitively, our model learns to skip words that do not need to be modified by predicting `KEEP`, so it can focus on simplifying the parts that actually require changes. An analogy can be drawn to residual connections popular in deep neural archi-

---

[1]Link to our code and data can be found here `https://github.com/yuedongP/EditNTS`.

WikiLarge		
Source	in 2005 , meissner became the second american woman to land the triple axel jump in national competition .	
Output	meissner was the second american woman to land the triple axel jump .	
Program	DEL DEL DEL KEEP ADD(was) DEL KEEP KEEP KEEP KEEP KEEP KEEP KEEP KEEP KEEP KEEP DEL DEL DEL KEEP	
Reference	she is the second american woman and the sixth woman worldwide to do a triple axel jump .	

WikiSmall		
Source	theodoros " thodoris " zagorakis -lrb- , born october 27 , 1971 in lyd -lrb- a village near the city of kavala -rrb- , is a retired greek footballer and was the captain of the greece national football team that won the 2004 uefa european football championship .	
Output	zagorakis -lrb- born october 27 , 1971 is a former greek football player .	
Program	DEL DEL DEL DEL KEEP KEEP DEL KEEP KEEP KEEP KEEP KEEP DEL DEL DEL DEL DEL DEL DEL DEL DEL DEL DEL DEL KEEP KEEP ADD(former) DEL KEEP ADD(football) ADD(player) DEL DEL ... DEL KEEP	
Reference	theodoros zagorakis -lrb- born 27 october , 1971 -rrb- is a former football player .	

Newsela		
Source	schools and parent groups try to help reduce costs for low-income students who demonstrate a desire to play sports , she said .	
Output	schools and parent groups try to help pay for low-income students .	
Program	KEEP KEEP KEEP KEEP KEEP KEEP KEEP ADD(pay) DEL DEL KEEP KEEP KEEP DEL DEL DEL DEL DEL DEL DEL DEL DEL DEL KEEP	
Reference	clark said that schools do sometimes lower fees for students who do n't have enough money .	

Table 1: Example outputs of EditNTS taken from the validation set of three text simplification benchmarks. Given a complex source sentence, our trained model predicts a sequence of edit tokens (EditNTS programs) that executes into a sequence of simplified tokens (EditNTS output).

tectures for image recognition, which give models the flexibility to directly copy parameters from previous layers if they are not the focus of the visual signal (He et al., 2016). In addition, the edit operations generated by our model are easier to interpret than the black-box MT-based seq2seq systems: by looking at our model's generated programs, we can trace the simplification operations used to transform complex sentences to simple ones. Moreover, our model offers control over the ratio of simplification operations. By simply changing the loss weights on edit operations, our model can prioritize different simplification operations for different sentence simplification tasks (e.g., compression or lexical replacement).

The idea of learning sentence simplification through edit operations was attempted by Alva-Manchego et al. (2017). They were mainly focused on creating better-aligned simplification edit labels ("silver" labels) and showed that a simple sequence labelling model (BiLSTM) fails to predict these silver simplification labels. We speculate that the limited success of their proposed model is due to the facts that the model relies on an external system and assumes the edit operations are independent of each other. We address these two problems by 1) using variants of Levenshtein distances to create edit labels that do not require

external tools to execute; 2) using an interpreter to execute the programs and summarize the partial output sequence immediately before making the next edit decision. Our interpreter also acts as a language model to regularize the operations that would lead to ungrammatical outputs, as a programmer alone will output edit labels with little consideration of context and grammar. In addition, our model is completely end-to-end and does not require any extra modules.

Our contributions are two-fold: 1) we propose to model the edit operations *explicitly* for sentence simplification in an end-to-end fashion, rather than relying on MT-based models to learn the simplification mappings implicitly, which often generates outputs by blindly repeating the source sentences; 2) we design an NPI-based model that simulates the editing process by a programmer and an interpreter, which outperforms the state-of-the-art neural MT-based TS models by large margins in terms of SARI and is judged by humans as simpler and overall better.

2 Related Work

MT-based Sentence Simplification SMT-based models and NMT-based models have been the main approaches for sentence simplification. They rely on learning simplification rewrites *implic-*

itly from complex-simple sentence pairs. For SMT-based models, Zhu et al. (2010) adopt a tree-based SMT model for sentence simplification; Woodsend and Lapata (2011) propose a quasi-synchronous grammar and use integer linear programming to score the simplification rules; Wubben et al. (2012) employ a phrase-based MT model to obtain candidates and re-rank them based on the dissimilarity to the complex sentence; Narayan and Gardent (2014) develop a hybrid model that performs sentence splitting and deletion first and then re-rank the outputs similar to Wubben et al. (2012); Xu et al. (2016) propose SBMT-SARI, a syntax-based machine translation framework that uses an external knowledge base to encourage simplification. On the other side, many NMT-based models have also been proposed for sentence simplification: Nisioi et al. (2017) employ vanilla recurrent neural networks (RNNs) on text simplification; Zhang and Lapata (2017) propose to use reinforcement learning methods on RNNs to optimize a specific-designed reward based on simplicity, fluency and relevancy; Vu et al. (2018) incorporate memory-augmented neural networks for sentence simplification; Zhao et al. (2018) integrate the transformer architecture and PPDB rules to guide the simplification learning; Sulem et al. (2018b) combine neural MT models with sentence splitting modules for sentence simplification.

Edit-based Sentence Simplification The only previous work on sentence simplification by *explicitly* predicting simplification operations is by Alva-Manchego et al. (2017). Alva-Manchego et al. (2017) use MASSAlign (Paetzold et al., 2017) to obtain 'silver' labels for simplification edits and employ a BiLSTM to sequentially predict three of their silver labels—KEEP, REPLACE and DELETE. Essentially, their labelling model is a non-autoregressive classifier with three classes, where a downstream module (Paetzold and Specia, 2017) is required for applying the REPLACE operation and providing the replacement word. We instead propose an end-to-end neural programmer-interpreter model for sentence simplification, which does not rely on external simplification rules nor alignment tools[2].

Neural Programmer-Interpreter Models The neural programmer-interpreter (NPI) was first proposed by Reed and de Freitas (2016) as a machine learning model that learns to execute programs given their execution traces. Their experiments demonstrate success for 21 tasks including performing addition and bubble sort. It was adopted by Ling et al. (2017) to solve algebraic word problems and by Bérard et al. (2017); Vu and Haffari (2018) to perform automatic post-editing on machine translation outputs. We instead design our NPI model to take monolingual complex input sentences and learn to perform simplification operations on them.

3 Model

Conventional sequence-to-sequence learning models map a sequence $\mathbf{x} = x_1, \ldots, x_{|\mathbf{x}|}$ to another one $\mathbf{y} = y_1, \ldots, y_{|\mathbf{y}|}$, where elements of $\mathbf{x}$ and $\mathbf{y}$ are drawn from a vocabulary of size V, by modeling the conditional distribution $P(y_t|y_{1:t-1}, \mathbf{x})$ directly. Our proposed model, EditNTS, tackles sentence simplification in a different paradigm by learning the simplification operations explicitly. An overview of our model is shown in Figure 1.

3.1 EditNTS Model

EditNTS frames the simplification process as executing a sequence of edit operations on complex tokens monotonically. We define the edit operations as {ADD(W), KEEP, DELETE, STOP}. Similar to the sequence-to-sequence learning models, we assume a fixed-sized vocabulary of V words that can be added. Therefore, the number of prediction candidates of the programmer is $V + 3$ after including KEEP, DELETE, and STOP. To solve the out-of-vocabulary (OOV) problem, conventional Seq2Seq models utilize a copy mechanism (Gu et al., 2016) that selects a word from source (complex) sentence directly with a trainable pointer. In contrast, EditNTS has the ability to copy OOV words into the simplified sentences by directly learning to predict KEEP on them in complex sentences. We argue that our method has advantage over a copy mechanism in two ways: 1) our method does not need extra parameters for copying; 2) a copy mechanism may lead to the model copying blindly rather than performing simplifications.

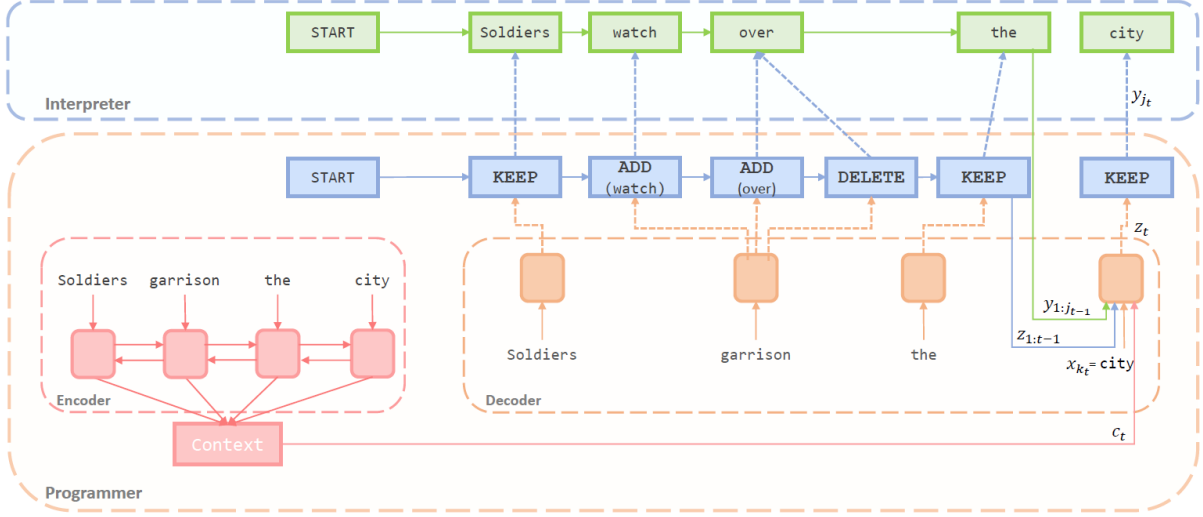We detail other constraints on the edit opera-

Figure 1: Our model contains two parts: the programmer and the interpreter. At time step t, the programmer predicts an edit operation z_t on the complex word x_{k_t} by considering the interpreter-generated words $y_{1:j_{t-1}}$, programmer-generated edit labels $z_{1:t-1}$, and a context vector c_t obtained by attending over all words in the complex sentence. The interpreter executes the edit operation z_t to generate the simplified token y_{j_t} and provides the interpreter context $y_{1:j_t}$ to the programmer for the next decision.

tions in Section 3.2. It turns out that the sequence of edit operations $\mathbf{z}$ constructed by Section 3.2 is deterministic given $\mathbf{x}$ and $\mathbf{y}$ (an example of of $\mathbf{z}$ can be seen in Table 2). Consequently, EditNTS can learn to simplify by modelling the conditional distribution $P(\mathbf{z}|\mathbf{x})$ with a programmer, an interpreter and an edit pointer:

$$P(\mathbf{z}|\mathbf{x}) = \prod_{t=1}^{\mathbf{z}} P(z_t|y_{1:j_{t-1}}, z_{1:t-1}, x_{k_t}, \mathbf{x}). \quad (1)$$

Complex sentence $\mathbf{x} = x_1, \dots x_{|\mathbf{x}|}$
['the', 'line', 'between', 'combat', 'is', 'getting', 'blurry']
Simple sentence $\mathbf{y} = y_1, \dots y_{|\mathbf{y}|}$
['war', 'is', 'changing']
Supervised programs $\mathbf{z} = z_1, \dots, z_{|\mathbf{z}|}$
[ADD('war'), DEL, DEL, DEL, DEL, KEEP, ADD('changing'), DEL, DEL]

Table 2: Given the source sentence $\mathbf{x}$ and the target sentence $\mathbf{y}$, our label creation algorithm (section 3.2) generates a deterministic program sequence $\mathbf{z}$ for training.

At time step t, the programmer decides an edit operation z_t on the word x_{k_t}, which is assigned by the edit pointer, based on the following contexts: 1) the summary of partially edited text $y_{1:j_{t-1}}$, 2) the previously generated edit operations $z_{1:t-1}$, 3) and the complex input sentence $\mathbf{x}$. The interpreter then executes the edit operation z_t into a simplified token y_{j_t} and updates the interpreter context based on $y_{1:j_t}$ to help the programmer at the next

time step. The model is trained to maximize Equation 1 where $\mathbf{z}$ is the expert edit sequence created in 3.2. We detail the components and functions of the programmer and the interpreter hereafter.

Programmer. The programmer employs an encoder-decoder structure to generate programs; i.e., sequences of edit operations $\mathbf{z}$. An encoder transforms the input sentence $\mathbf{x} = x_1, \dots x_{|\mathbf{x}|}$ into a sequence of latent representations h_i^{enc}. We additionally utilize the part-of-speech (POS) tags $\mathbf{g} = g_1, \dots g_{|\mathbf{x}|}$ to inject the syntactic information of sentences into the latent representations. The specific transformation process is:

$$h_i^{enc} = \text{LSTM}^{enc}([e_1(x_i), e_2(g_i)]) \quad (2)$$

where $e_1(\cdot)$ and $e_2(\cdot)$ are both look-up tables. The decoder is trained to predict the next edit label z_t (Eq. 3), given the vector representation $h_{k_t}^{enc}$ for the word x_{k_t} that currently needs to be edited (Eq. 2), vector representation h_t^{edit} of previously generated edit labels $z_{1:t-1}$ (Eq. 4), the source context vector c_t (Eq.5), and the vector representation of previously generated words by the interpreter $y_{1:j_{t-1}}$ (Eq. 6).

$$P_{edit} = \text{softmax}(V'(\tanh(V(h_t^{edit})) \quad (3)$$

$$h_t^{edit} = \text{LSTM}^{edit}([h_{k_t}^{enc}, c_t, h_{t-1}^{edit}, h_{t-1}^{int}]) \quad (4)$$

$$c_t = \sum_{j=1}^{|\mathbf{x}|} \alpha_{tj} h_j, \alpha_{tj} = \text{softmax}(h_{k_t}, h_j) \quad (5)$$

3396

Note that there are three attentions involved in the computation of the programmer. 1) the soft attention over all complex tokens to form a context c_t; 2) k_t: the hard attention over complex input tokens for the **edit pointer**, which determines the index position of the current word that needs to be edited at t. We force k_t to be the number of KEEP and DELETE previously predicted by the programmer up to time t. 3) j_{t-1}: the hard attention over simple tokens for training (this attention is used to speed up the training), which is the number of KEEP and ADD(W) in the reference gold labels up to time $t-1$. During inference, the model no longer needs this attention and instead incrementally obtains $y_{1:j_{t-1}}$ based on its predictions.

Interpreter. The interpreter contains two parts: 1) a parameter-free executor $exec(z_t, x_{k_t})$ that applies the predicted edit operation z_t on word x_{k_t}, resulting in a new word y_{j_t}. The specific execution rules for the operations are as follows: execute KEEP/DELETE to keep/delete the word and move the edit pointer to the next word; execute ADD(W) to add a new word W and the edit pointer stays on the same word; and execute STOP to terminate the edit process. 2) an LSTM interpreter (Eq. 6) that summarizes the partial output sequence of words produced by the executor so far. The output of the LSTM interpreter is given to the programmer in order to generate the next edit decision.

$$h_t^{int} = \text{LSTM}^{int}([h_{t-1}^{int}, y_{j_{t-1}}]) \qquad (6)$$

3.2 Edit Label Construction

Unlike neural seq2seq models, our model requires expert programs for training. We construct these expert edit sequences from complex sentences to simple ones by computing the shortest edit paths using a dynamic programming algorithm similar to computing Levenshtein distances without substitutions. When multiple paths with the same edit distance exist, we further prioritizes the path that ADD before DELETE. By doing so, we can generate a unique edit path from a complex sentence to a simple one, reducing the noise and variance that the model would face [3]. Table 2 demonstrates an example of the created edit label path and Table 3 shows the counts of the created edit labels

---

[3] We tried other way of labelling, such as 1) preferring DELETE to ADD; 2) deciding randomly when there is a tie; 3) including REPLACE as an operation. However, models trained with these labelling methods do not give good results from our empirical studies.

on the training sets of the three text simplification corpora.

	KEEP	DELETE	ADD	STOP
WikiLarge	2,781,648	3,847,848	2,082,184	246,768
WikiSmall	1,356,170	780,482	399,826	88,028
Newsela	1,042,640	1,401,331	439,110	94,208

Table 3: Counts of the edit labels constructed by our label edits algorithm on three dataset (identical complex-simple sentence pairs are removed).

As can be seen from Table 3, our edit labels are very imbalanced, especially on DELETE. We resolve this by two approaches during training: 1) we associate the inverse of edit label frequencies as the weights to calculate the loss; 2) the model only executes DELETE when there is an explicit DELETE prediction. Thus, if the system outputs STOP before finish editing the whole complex sequence, our system will automatically pad KEEP until the end of the sentence, ensuring the system outputs remain conservative with respect to the complex sequences.

4 Experiments

4.1 Dataset

Three benchmark text simplification datasets are used in our experiments. **WikiSmall** contains automatically aligned complex-simple sentence pairs from standard to simple English Wikipedia (Zhu et al., 2010). We use the standard splits of 88,837/205/100 provided by Zhang and Lapata (2017) as train/dev/test sets. **WikiLarge** (Zhang and Lapata, 2017) is the largest TS corpus with 296,402/2000/359 complex-simple sentence pairs for training/validating/testing, constructed by merging previously created simplification corpora (Zhu et al., 2010; Woodsend and Lapata, 2011; Kauchak, 2013). In addition to the automatically aligned references, Xu et al. (2016) created eight more human-written simplified references for each complex sentence in the development/test set of WikiLarge. The third dataset is **Newsela** (Xu et al., 2015), which consists of 1130 news articles. Each article is rewritten by professional editors four times for children at different grade levels (0-4 from complex to simple). We use the standard splits provided by Zhang and Lapata (2017), which contains 94,208/1129/1076 sentence pairs for train/dev/test. Table 4 provides other statistics on these three benchmark training

sets.

	Vocabulary size		Sentence length	
	comp	simp	comp	simp
WikiLarge	201,841	168,962	25.17	18.51
WikiSmall	113,368	93,835	24.26	20.33
Newsela	41,066	30,193	25.94	15.89

Table 4: Statistics on the vocabulary sizes and the average sentence lengths of the complex and simplified sentences in the three text simplification training sets.

4.2 Baselines

We compare against three state-of-the-art SMT-based TS systems: PBMT-R (Wubben et al., 2012) where the phrase-based MT system's outputs are re-ranked; 2) Hybrid (Narayan and Gardent, 2014) where syntactic transformation such as sentence splits and deletions are performed before re-rank; 3) SBMT-SARI (Xu et al., 2016), a syntax-based MT framework with external simplification rules. We also compare against four state-of-the-art NMT-based TS systems: vanilla RNN-based model NTS (Nisioi et al., 2017), memory-augmented neural networks $N_{SE}L_{STM}$ (Vu et al., 2018), deep reinforcement learning-based neural network DRESS and DRESS-LS (Zhang and Lapata, 2017), and DMASS+DCSS (Zhao et al., 2018) that integrates the transformer model with external simplification rules. In addition, we compare our NPI-based EditNTS with the BiLSTM sequence labelling model (Alva-Manchego et al., 2017) that are trained on our edit labels[4], we call it Seq-Label model.

4.3 Evaluation

We report two widely used sentence simplification metrics in the literature: SARI (Xu et al., 2016) and FKGL (Kincaid et al., 1975). FKGL (Kincaid et al., 1975) measures the readability of the system output (lower FKGL implies simpler output) and SARI (Xu et al., 2016) evaluates the system output by comparing it against the source and reference sentences. Earlier work also used BLEU as a metric, but recent work has found that it does not reflect simplification (Xu et al., 2016) and is in fact negatively correlated with simplicity (Sulem et al., 2018a). Systems with high BLEU scores are thus

biased towards copying the complex sentence as a whole, while SARI avoids this by computing the arithmetic mean of the N-gram ($N \in \{1, 2, 3, 4\}$) F1-scores of three rewrite operations: add, delete, and keep. We also report the F1-scores of these three operations. In addition, we report the percentage of unchanged sentences that are directly copied from the source sentences. We treat SARI as the most important measurement in our study, as Xu et al. (2016) demonstrated that SARI has the highest correlation with human judgments in sentence simplification tasks.

In addition to automatic evaluations, we also report human evaluations[5] of our system outputs compared to the best MT-based systems, external knowledge-based systems, and Seq-Label by three human judges[6] with a five-point Likert scale. The volunteers are asked to rate simplifications on three dimensions: 1) fluency (is the output grammatical?), 2) adequacy (how much meaning from the original sentence is preserved?), and 3) simplicity (is the output simper than the original sentence?).

4.4 Training Details

We used the same hyperparameters across the three datasets. We initialized the word and edit operation embeddings with 100-dimensional GloVe vectors (Pennington et al., 2014) and the part-of-speech tag [7] embeddings with 30 dimensions. The number of hidden units was set to 200 for the encoder, the edit LSTM, and the LSTM interpreter. During training, we regularized the encoder with a dropout rate of 0.3 (Srivastava et al., 2014). For optimization, we used Adam (Kingma and Ba, 2014) with a learning rate 0.001 and weight decay of 10^{-6}. The gradient was clipped to 1 (Pascanu et al., 2013). We used a vocabulary size of 30K and the remaining words were replaced with UNK. In our main experiment, we used the inverse

---

[4]We made a good faith reimplementation of their model and trained it with our created edit labels. We cannot directly compare with their results because their model is not available and their results are not obtained from standard splits.

[5]The outputs of PBMT-R, Hybrid, SBMT-SARI and DRESS are publicly available and we are grateful to Sanqiang Zhao for providing their system's outputs.

[6]Three volunteers (one native English Speaker and two non-native fluent English speakers) are participated in our human evaluation, as one of the goal of our system is to make the text easier to understand for non-native English speakers. The volunteers are given complex setences and different system outputs in random order, and are asked to rate from one to five (the higher the better) in terms of simplicity, fluency, and adequacy.

[7]We used the NLTK toolkit with the default Penn Treebank Tag set to obtain the part-of-speech tags; there are 45 possible POS-tags (36 standard tags and 7 special symbols) in total.

of the edit label frequencies as the loss weights, aiming to balance the classes. Batch size across all datasets was 64.

5 Results

WikiLarge	SARI	Edit F1 of SARI			FKGL	% unc.
		add	del	keep		
Reference	-	-	-	-	8.88	15.88
MT-based TS Models						
PBMT-R	**38.56**	**5.73**	36.93	73.02	8.33	**10.58**
Hybrid	31.40	1.84	**45.48**	46.87	**4.57**	36.21
NTS	35.66	2.99	28.96	**75.02**	8.42	43.45
N$_{SE}$L$_{STM}$	36.88	-	-		- -	
DRESS	37.08	2.94	43.15	65.15	6.59	22.28
DRESS-LS	37.27	2.81	42.22	66.77	6.62	27.02
Edit Labelling-based TS Models						
Seq-Label	37.08	2.94	43.20	65.10	5.35	19.22
EditNTS	38.22	3.36	39.15	72.13	7.30	10.86
Models that use external knowledge base						
SBMT-SARI	39.96	5.96	41.42	72.52	7.29	9.47
DMASS+DCSS	40.45	5.72	42.23	73.41	7.79	6.69

(a) WikiLarge

WikiSmall	SARI	Edit F1 of SARI			FKGL	% unc.
		add	del	keep		
Reference	-	-	-	-	8.86	3.00
MT-based TS Models						
PBMT-R	15.97	6.75	28.50	12.67	11.42	14.00
Hybrid	30.46	**16.53**	59.60	**15.25**	9.20	4.00
NTS	13.61	2.08	26.21	12.53	11.35	36.00
N$_{SE}$L$_{STM}$	29.75	-	-	-	-	-
DRESS	27.48	2.86	65.94	13.64	7.48	11.00
DRESS-LS	27.24	3.75	64.27	13.71	7.55	13.00
Edit Labelling-based TS Models						
Seq-Label	30.50	2.72	76.31	12.46	9.38	9.00
EditNTS	**32.35**	2.24	**81.30**	13.54	**5.47**	**0.00**

(b) WikiSmall

Newsela	SARI	Edit F1 of SARI			FKGL	%unc.
		add	delete	keep		
Reference	-	-	-	-	3.20	0.00
MT-based TS Models						
PBMT-R	15.77	3.07	38.34	5.90	7.59	5.85
Hybrid	30.00	1.16	83.23	5.62	4.01	**3.34**
NTS	24.12	2.73	62.66	6.98	5.11	16.25
N$_{SE}$L$_{STM}$	29.58	-	-	-	-	-
DRESS	27.37	3.08	71.61	**7.43**	4.11	11.98
DRESS-LS	26.63	**3.21**	69.28	7.40	4.20	15.51
Edit Labelling-based TS Models						
Seq-Label	29.53	1.40	80.25	6.94	5.45	15.97
EditNTS	**31.41**	1.84	**85.36**	7.04	**3.40**	4.27

(c) Newsela

Table 5: Automatic Evaluation Results on three benchmarks. We report corpus level FKGL, SARI and edit F1 scores (add,keep,delete). In addition, we report the percentage of unchanged sentences (%unc.) in the system outputs when compared to the source sentences.

Table 5 summarizes the results of our automatic evaluations. In terms of readability, our system obtains lower (= better) FKGL compared to other MT-based systems, which indicates our system's output is easier to understand. In terms of the percentage of unchanged sentences, one can see that MT-based models have much higher rates of unchanged sentences than the reference. Thus, the models learned a safe but undesirable strategy of copying the sources sentences directly. By contrast, our model learns to edit the sentences and has a lower rate of keeping the source sentences unchanged.

In term of SARI, the edit labelling-based models Seq-Label and EditNTS achieve better or comparable results with respect to state-of-the-art MT-based models, demonstrating the promise of learning edit labels for text simplification. Compared to Seq-Label, our model achieves a large improvement of (+1.14,+1.85,+1.88 SARI) on WikiLarge, Newsela, and WikiSmall. We believe this improvement is mainly from the interpreter in EditNTS, as it provides the proper context to the programmer for making edit decisions (more ablation studies in section 5.1). On Newsela and WikiSmall, our model significantly outperforms state-of-the-art TS models by a large margin (+1.89, +1.41 SARI), showing that EditNTS learns simplification better on smaller datasets with respect to MT-based simplification models. On WikiLarge, our model outperforms the best NMT-based system DRESS-LS by a large margin of +0.95 SARI and achieves comparable performance to the best SMT-based model PBMT-R. While the overall SARI are similar between EditNTS and PBMT-R, the two models prefer different strategies: EditNTS performs extensive `DELETE` while PBMT-R is in favour of performing lexical substitution and simplification.

On WikiLarge, two models SBMT-SARI and DMASS+DCSS reported higher SARI scores as they employ external knowledge base PPDB for word replacement. These external rules can provide reliable guidance about which words to modify, resulting in higher add/keep F1 scores (Table 5-a). On the contrary, our model is inclined to generate shorter sentences, which leads to high F1 scores on delete operations [8]. Nevertheless, our model is preferred by human judges than SBMT-

---

[8] As the full outputs of N$_{SE}$L$_{STM}$ are not available, we cannot compute the edit F1 scores and FKGL for this system.

	WikiLarge				Newsela				WikiSmall			
	F	A	S	avg.	F	A	S	avg.	F	A	S	avg.
Reference	4.39	4.11	2.62	3.71	4.40	2.74	3.79	3.64	4.48	4.03	2.99	3.83
PBMT-R	4.38	4.05	2.28	3.57	3.76	**3.44**	2.28	3.16	4.32	**4.28**	1.53	3.38
Hybrid	3.41	3.01	**3.31**	3.24	3.62	2.88	2.97	3.16	3.76	3.87	2.12	3.25
SBMT-SARI	4.25	3.96	2.61	3.61	-	-	-	-	-	-	-	-
DRESS	4.63	4.01	3.07	3.90	4.16	3.08	3.00	3.41	**4.61**	3.64	3.62	3.96
DMASS+DCSS	4.39	3.97	2.80	3.72	-	-	-	-	-	-	-	-
seq-label	3.91	4.11	2.97	3.66	3.45	3.22	2.09	2.92	3.83	3.9	2.01	3.25
EditNTS	**4.76**	**4.45**	3.18	**4.13**	**4.34**	3.13	**3.16**	**3.54**	4.31	3.34	**4.26**	**3.97**

Table 6: Mean ratings for Fluency (F), Adequacy (A), Simplicity (S), and the Average score (avg.) by human judges on the three benchmark test sets. 50 sentences are rated on WikiLarge, 30 sentences are rated on WikiSmall and Newsela. Aside from comparing system outputs, we also include human ratings for the gold standard reference as an upper bound.

SARI and DMASS+DCSS in terms of all the measurements (Table 6), indicating the effectiveness of our model on correctly performing deleting operations while maintaining fluent and adequate outputs. Moreover, our model can be easily integrated with these external PPTB simplification rules for word replacement by adding a new edit label "replacement" for further improvements.

The results of our human evaluations are presented in Table 6. As can be seen, our model outperforms MT-based models on Fluency, Simplicity, and Average overall ratings. Despite our system EditNTS is inclined to perform more delete operations, human judges rate our system as adequate. In addition, our model performs significantly better than Seq-Label in terms of Fluency, indicating the importance of adding an interpreter to 1) summarize the partial edited outputs and 2) regularize the programmer as a language model. Interestingly, similar to the human evaluation results in Zhang and Lapata (2017), judges often prefer system outputs than the gold references.

Controllable Generation: In addition to the state-of-the-art performance, EditNTS has the flexibility to prioritize different edit operations. Note that NMT-based systems do not have this feature at all, as the sentence length of their systems' output is not controllable and are purely depends on the training data. Table 7 shows that by simply changing the loss weights on different edit labels, we can control the length of system's outputs, how much words it copies from the original sentences and how much novel words the system adds.

5.1 Ablation Studies

In the ablation studies, we aim to investigate the effectiveness of each component in our model. We

add:keep:delete ratio	Avg. len	% copied	% novel
10:1:1 (add rewarded)	25.21	53.52	56.28
1:10:1 (keep rewarded)	21.52	84.22	12.81
1:1:10 (delete rewarded)	15.83	57.36	16.72

Table 7: Results on Newsela by controlling the edit label ratios. We increase the loss weight on ADD,KEEP,DELETE ten times respectively. The three rows show the systems' output statistics on the average output sentence length (Avg. len), the average percentage of tokens that are copied from the input (% copied), and the average percentage of novel tokens that are added with respect to the input sentence (% novel).

compare the full model with its variants where POS tags removed, interpreter removed, context removed. As shown in Table 8, the interpreter is a critical part to guarantee the performance of the sequence-labelling model, while POS tags and attention provide further performance gains.

Newsela	SARI	Edit F1 of SARI		
		add	delete	keep
EditNTS	31.41	1.84	85.36	7.04
— POS tags	31.27	1.46	85.34	7.00
— attn-context	30.95	1.54	84.26	7.05
— Interpreter	30.13	1.70	81.70	7.01

Table 8: Performance on Newsela after removing different components in EditNTS.

6 Conclusion

We propose an NPI-based model for sentence simplification, where edit-labels are predicted by the programmer and then executed into simplified tokens by the interpreter. Our model outperforms previous state-of-the-art machine translation-based TS models in most of the au-

tomatic evaluation metrics and human ratings, demonstrating the effectiveness of learning edit operations *explicitly* for sentence simplification. Compared to the black-box MT-based systems, our model is more interpretable by providing generated edit operation traces, and more controllable with the ability to prioritize different simplification operations.

Acknowledgments

References

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)*, volume 1, pages 295–305.

Alexandre Bérard, Laurent Besacier, and Olivier Pietquin. 2017. Lig-cristal submission for the wmt 2017 automatic post-editing task. In *Proceedings of the Second Conference on Machine Translation*, pages 623–629.

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*.

Richard Evans, Constantin Orasan, 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.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor OK Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. *arXiv preprint arXiv:1603.06393*.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

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)*, volume 1, pages 1537–1546.

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. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Wang Ling, Dani Yogatama, Chris Dyer, and Phil Blunsom. 2017. Program induction by rationale generation: Learning to solve and explain algebraic word problems. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 158–167.

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)*, volume 1, pages 435–445.

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)*, volume 2, pages 85–91.

Gustavo Paetzold, Fernando Alva-Manchego, and Lucia Specia. 2017. Massalign: Alignment and annotation of comparable documents. *Proceedings of the IJCNLP 2017, System Demonstrations*, pages 1–4.

Gustavo Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, volume 2, pages 34–40.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. 2013. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543.

Scott Reed and Nando de Freitas. 2016. Neural programmer-interpreters. In *Proceedings of International Conference on Learning Representations (ICLR)*.

Luz Rello, Clara Bayarri, Azuki Górriz, Ricardo Baeza-Yates, Saurabh Gupta, Gaurang Kanvinde, Horacio Saggion, Stefan Bott, Roberto Carlini, and

Vasile Topac. 2013. Dyswebxia 2.0!: more accessible text for people with dyslexia. In *Proceedings of the 10th International Cross-Disciplinary Conference on Web Accessibility*, page 25. Citeseer.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *The Journal of Machine Learning Research*, 15(1):1929–1958.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. 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.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Simple and effective text simplification using semantic and neural methods. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, volume 1, pages 162–173.

Thuy-Trang Vu and Gholamreza Haffari. 2018. Automatic post-editing of machine translation: A neural programmer-interpreter approach. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3048–3053.

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)*, volume 2, pages 79–85.

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*, pages 29–36. ACM.

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. Association for Computational Linguistics.

Sander Wubben, Antal Van Den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1015–1024. Association for Computational Linguistics.

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

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

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.

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

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

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*, pages 1353–1361. Association for Computational Linguistics.