Learning How to Simplify From Explicit Labeling of Complex-Simplified Text Pairs

Fernando Alva-Manchego[1] and **Joachim Bingel**[2] and **Gustavo H. Paetzold**[1]
Carolina Scarton[1] and **Lucia Specia**[1]
[1]Department of Computer Science, University of Sheffield, UK
[2]Department of Computer Science, University of Copenhagen, Denmark
{f.alva,g.h.paetzold,c.scarton,l.specia}@sheffield.ac.uk
bingel@di.ku.dk

Abstract

Current research in text simplification has been hampered by two central problems: (i) the small amount of high-quality parallel simplification data available, and (ii) the lack of explicit annotations of simplification operations, such as deletions or substitutions, on existing data. While the recently introduced Newsela corpus has alleviated the first problem, simplifications still need to be learned directly from parallel text using black-box, end-to-end approaches rather than from explicit annotations. These complex-simple parallel sentence pairs often differ to such a high degree that generalization becomes difficult. End-to-end models also make it hard to interpret what is actually learned from data. We propose a method that decomposes the task of TS into its sub-problems. We devise a way to automatically identify operations in a parallel corpus and introduce a sequence-labeling approach based on these annotations. Finally, we provide insights on the types of transformations that different approaches can model.

1 Introduction

Text Simplification (TS) is the task of reducing the complexity of a text without changing its meaning. Simplification can be applied at various linguistic levels, from lexical substitution to more global operations such as sentence splitting, paraphrasing or the deletion or reordering of entire clauses.

Existing corpora for TS generally come in one of two variants. The first focuses on very specific sub-problems, such as sentence compression (Bingel and Søgaard, 2016) or the identification of difficult words (Paetzold and Specia, 2016a), and typically encodes relevant simplification operations as discrete labels on tokens. The other variant includes more general, higher-level types of simplifications that often entail the rephrasing or re-structuring of sentences, with content added or removed. These "natural" simplifications are often created for end-users rather than for research purposes. Examples of the latter simplification resources include the Newsela (Xu et al., 2015) and Simple English Wikipedia corpora (Zhu et al., 2010; Coster and Kauchak, 2011b). These resources generally encode interdependencies between different types of simplification better than single-purpose resources and may thus seem favorable for learning simplifications. However, the freedom given to editors and lack of explicit labels on the modifications performed makes generalization much more difficult, especially when existing resources are relatively small in comparison to corpora for other text-to-text problems like machine translation (MT). Nevertheless, these corpora have been extensively used to learn phrase-based statistical and neural models for end-to-end TS systems that bear resemblance to MT models (Specia, 2010; Zhu et al., 2010; Coster and Kauchak, 2011b; Wubben et al., 2012; Narayan and Gardent, 2014; Xu et al., 2016; Zhang and Lapata, 2017; Zhang et al., 2017; Nisioi et al., 2017).

Adaptability and interpretability MT-style models are essentially black boxes that offer little or no control over the way in which a given input is modified. Additionally, in most cases the types of modifications that are actually learned are limited to paraphrasing of short sequences of words. We believe a middle ground is missing in terms of resources and approaches for TS, where models are learned from a more informed labeled

FA and JB contributed equally to this paper.

dataset of natural simplifications, and can then be applied in a controlled way, e.g., in adaptive simplification scenarios that prioritize different ways of simplifying (e.g. compression or sentence splitting) depending on a particular user's needs.

The only previous work on TS via explicitly predicting simplification operations is that by Bingel and Søgaard (2016), who create training data from comparable text to label entire syntactic units and train a sequence labeling model to predict deletions and phrase substitutions in a complex sentence. Our approach is different in that it captures a larger variety of operations in a more global fashion, by using sentence-wide word alignments rather than surface heuristics. Furthermore, we use a more reliable (professionally created) corpus and our approach is more flexible as we do not rely on syntactic parse trees at test time.

Contributions This paper introduces the following main contributions: **(1)** We provide an in-depth analysis on the potential and limitations of the dominant approach to TS: end-to-end MT-style models; **(2)** We devise a method to automatically identify specific simplification operations in aligned sentences from complex-to-simple simplification corpora. This results in a corpus that can be used to study how human experts perform simplification tasks, as well as to train simplification models to address specific problems; and **(3)** We propose a sequence labeling model built from such a corpus to predict which simplification operations should be performed as a first step for a complete simplification pipeline. This approach is highly modular: once operations are identified, different methods can be applied to cover each simplification operation. We show that this operation-based TS approach is able to produce simpler texts than end-to-end models. The code for extracting the simplification operations is available at `https://github.com/ghpaetzold/massalign`, while our sequence labeling model is released at `https://github.com/jbingel/ijcnlp2017_simplification`.

2 Related Work

In what follows we give a brief description of previous work on statistical and neural models for TS. We first compare methods using versions of Simple English Wikipedia data (Zhu et al., 2010; Coster and Kauchak, 2011b), before considering recent work that relies on the professionally edited Newsela corpus (Xu et al., 2015).

Simple English Wikipedia Zhu et al. (2010) propose a syntax-based translation model for TS that learns operations over the parse trees of the complex sentences. They outperform several baselines in terms of Flesch index. Coster and Kauchak (2011b) train a phrase-based machine translation (PBMT) system and obtain significant improvements in terms of BLEU (Papineni et al., 2002) over a baseline. Coster and Kauchak (2011a) extend a PBMT model to include phrase deletion and outperform Coster and Kauchak (2011b). Wubben et al. (2012) also train a PBMT system for TS with a dissimilarity-based re-ranking heuristic, outperforming Zhu et al. (2010) in terms of BLEU. Narayan and Gardent (2014) built TS systems by combining discourse representation structures with a PBMT model, which outperforms previous approaches. Xu et al. (2016) modify a syntax-based MT system in order to use a new metric – SARI – for optimization and to include special rules for paraphrasing. Although their system does not outperform previous work in terms of BLEU, it achieves the best results according to SARI and human evaluation. Zhang et al. (2017) train a lexically constrained sequence-to-sequence neural network model for TS, based on the encoder-decoder architecture for MT. The system outperforms baseline systems (including a PBMT system) in terms of BLEU. Finally, Nisioi et al. (2017) propose a model for TS that is able to perform lexical replacements and content reduction. They use a neural encoder-decoder approach where they combine pre-trained (general domain and in-domain) word embeddings for the source and target sentences. They also perform beam search, finding the best beam size using either BLEU or SARI. Their best model outperforms previous PBMT-based approaches in terms of BLEU.

Newsela corpus To the best of our knowledge, Zhang and Lapata (2017) is the only work that explores MT-based approaches on the Newsela corpus. They train an attention-based encoder-decoder model (Bahdanau et al., 2014) and use reinforcement learning with a reward policy combining SARI, BLEU and cosine similarity (to measure meaning preservation). Their approach shows improvements over a PBMT system in terms of

BLEU and SARI, but no insights are given with respect to the transformations that are actually learned or how distant from the original sentences the simplifications are. They also experiment with the Simple Wikipedia corpus, yet do not outperform Narayan and Gardent (2014) on this data.

The neural end-to-end model we implement as a baseline in this paper is equivalent to that in Zhang et al. (2017) without the lexical constraints, while the statistical model is equivalent to the one in Coster and Kauchak (2011b).

3 Simplification via End-to-End Models

In addition to requiring large amounts of training data, MT-based approaches to TS are limited because of their black-box way of addressing the problem. As we are going to show in this section, standard end-to-end systems without special adaptation to TS do not succeed in learning alternative formulations of the original text. With a few exceptions (by the neural model), they tend to repeat the original text. We conjecture that this is because, for most original-side material, TS corpora do not consistently enough offer alternative simplified formulations: in the majority of instances, most words are kept as in the original.

To study the potential and limitations of end-to-end translation models for TS, we build models using state-of-the-art MT-based approaches and the Newsela corpus, arguably the most reliable (professionally created) and realistic (aimed at a target audience rather than research) resource to date.

The Newsela Corpus.[1] Newsela is a multi-comparable corpus where each document comes in up to six levels of simplicity, from 0 (original) to 5 (simplest). In our experiments, we only use sentence pairs stemming from adjacent levels of simplicity within the same document.[2]

Translation approaches require data aligned at the sentence level. Given the original Newsela corpus, which only aligns different versions of the same document, we first align sentences using the algorithms described in (Paetzold and Specia, 2016b). Their algorithms search for the best alignment path between the paragraphs and sentences of parallel documents based on TF-IDF cosine similarity and an incremental vicinity search range. They address limitations of previous strategies (Barzilay and Elhadad, 2003; Coster and Kauchak, 2011b; Smith et al., 2010; Xu et al., 2015; Bott and Saggion, 2011) by disregarding the need for (semi-) supervised training, allowing long-distance alignment skips, and capturing N-to-N alignments. The alignments produced are categorized as:

- **Identical:** The alignment is one-to-one and the sentences are exactly the same (96,909 pairs across all adjacent levels).
- **1-to-1:** The alignment is one-to-one and the original-simplified sentences are different (130,790 pairs across all adjacent levels).
- **Split:** The alignment is 1-to-N (42,545 pairs across all adjacent levels).
- **Join:** The alignment is N-to-1 (7,962 pairs across all adjacent levels).

Translation Models. We built two types of models using state-of-the-art MT-based approaches: a phrase-based statistical MT model using Moses (Koehn et al., 2007),[3] and a neural MT model using Nematus (Sennrich et al., 2017).[4] The Neural Text Simplification tool (NTS) made available by Nisioi et al. (2017) was also used for comparison.[5]

For our translation-based experiments, we consider two combinations of sentence alignments, using (i) only one-to-one alignments (**1-to-1**) (130,970 sentence pairs), and (ii) all alignments (**all**), i.e., the entire sentence-aligned corpus with identical, 1-to-1, split and join alignments (278,206 sentence pairs). The first type of data (1-to-1) is the focus of this paper (see §4). The latter variant is included in the experiments for comparison, in particular to address the question whether more (but not necessarily better) data can aid data-intensive translation-based approaches. For all

[1] The Newsela Article Corpus was downloaded from https://newsela.com/data, version 2016-01-29.

[2] The motivations for only using adjacent levels are (i) that we assume that these are not "naturally" created (i.e. an expert would not start from an original text and directly generate a level 5 text, but rather go from 0 to 1, 1 to 2, ..., 4 to 5), and (ii) that the high degree of linguistic and stylistic differences between non-adjacent levels makes learning even more complex. For example, the average edit distance for sentences in the 0-1 group is 0.19, while for sentences in the 0-5 group, it is 0.65. As far as the first reason is concerned, note that we could not find any publicly available simplification guidelines for the Newsela corpus.

[3] We follow instructions from http://www.statmt.org/moses/?n=Moses.Baseline

[4] We use a vocabulary size of 30,000 and the same parameters as in Sennrich et al. (2016).

[5] We use the same configurations as Nisioi et al. (2017).

System	Hyp vs. Ref BLEU↑	Hyp vs. Ref TER↓	Hyp vs. Orig BLEU	Hyp vs. Orig TER	%Same↓	SARI↑
Moses (all)	69.64	30.20	98.77	0.41	93.03	27.45
Nematus (all)	36.46	52.66	45.40	42.30	21.60	22.91
NTS (all)	68.35	31.37	90.52	7.19	72.91	27.36
Moses (1-to-1)	57.79	40.19	98.30	0.86	89.50	24.58
Nematus (1-to-1)	46.90	52.84	76.29	20.10	30.45	29.89
NTS (1-to-1)	53.79	45.24	77.63	16.70	42.76	30.44
Silver operations (1-to-1)	67.33	22.66	61.63	26.01	10.83	61.71
Predicted operations (1-to-1)	41.37	48.72	59.71	25.24	14.06	31.29

Table 1: Performance of translation-based and operation-based TS models (using silver or predicted operation labels, with only DELETION and REPLACE applied). Metrics are BLEU and TER between simplified version (Hyp) and reference (Ref) or original version (Orig), the percentage of sentences copied from the input (%Same), and SARI for the simplifications.

experiments, the respectively used data was first randomly split into training (80%), development (10%) and test (10%) sets and normalized for entities (incl. names, locations, numbers).

Simplification Quality. The first and second sections of Table 1 show the results of translation-based systems according to several metrics: similarity metrics commonly used in MT, comprising BLEU (Papineni et al., 2002) and TER (Snover et al., 2006, minimum edit distance), as well a specific text simplification metric, SARI (Xu et al., 2016). SARI measures how good the words added, deleted and kept by a simplification system are, after comparing the produced output to the original sentence and the simplification reference(s). It is similar to BLEU but rewards copying words from the original sentence. According to experiments performed by Xu et al. (2016), SARI is the metric that best correlates with human judgments of simplicity.

For both "all" and "1-to-1" variants, the BLEU and TER scores between hypotheses and references are worse for Nematus, showing that a baseline neural model tends to be more aggressive and potentially generate noisier modifications than Moses equivalents. To measure how strongly the various approaches modify the input sentences, these scores are also reported between the generated simplifications and the original inputs. Again, these metrics are worse for Nematus-based models, showing that they indeed perform more modifications on the sentences. Moses in turn is very conservative, keeping 90-93% of the test sentences exactly in their original version. SARI shows low scores for all systems. NTS is also conservative in the "all" variant (attested by the high BLEU score between hypotheses and original sentences). For "1-to-1", NTS produces more simplifications, diverging more from the original sentences.

Sentence-level Operations. Interestingly, even though Moses and Nematus are trained on the same data, they differ substantially with respect to what they can learn. This is demonstrated by an automatic inspection we conducted on the simplifications produced by both systems trained over all types of sentence alignments, i.e. including sentence splits and joins.

Table 2 reports the count and proportion of instances in the test set representing types of sentence-level transformation between the original and simplified sentence. It can be noted that Moses is much more conservative than Nematus and simply tends to copy the original as the output ("Identical" cases). However, as the majority (57%) of aligned sentences in the professional Newsela simplifications are edited, we do not consider copying a valid "simplification" in most cases. Note also that Moses displays an excessively high BLEU score between the *original* and hypothesis sentences (98.77), while the similarity between the original and reference sentences is much lower (71.57).

Manually inspecting some of the simplifications made, we find that when it comes to sentence splits, both MT-based simplifiers seem to be able to perform this type of transformation in an accu-

	Moses		Nematus	
Operation	**Count**	**%**	**Count**	**%**
Identical	25,882	93.03	10,906	39.20
1-to-1	1,920	6.90	15,428	55.45
Split	14	0.05	354	1.27
Join	4	0.01	1,132	4.07

Table 2: Count and proportion of instances affected by each type of simplification transformation performed by Moses and Nematus.

rate way. However, the proportion of such cases is very low (0.05% and 1.27% for Moses and Nematus, respectively) compared to the proportion in the gold data (13.5%) of the sentence pairs contain at least one split.

Moses only joins sentences in four cases, but these are all spurious instances where a period is incorrectly removed. Nematus is more successful at learning this type of operation. In most cases, it discards entire clauses that contain less relevant content. For example, it simplifies the sentence "*Lincoln often cried in public and recited sad poetry, according to Joshusa Wolf Shenk, who wrote a book called Lincoln's Melancholy*" to "*Lincoln often cried in public and recited sad poetry*". We also find a few examples where the content that is not discarded is rewritten to some extent, mostly for grammaticality. The Nematus simplification of "*Frank was what the instructors called a 'rock star'; he emerged as a leader who worked hard to keep the group together*" onto "*Frank was a leader who worked hard to keep the group together* " is a good example of that.

When it comes to 1-to-1 transformations, which can include a number of different operations (see §4), most transformations made by Nematus consist of segment deletions, some of which are paired with localized segment rewritings. As for Moses, most 1-to-1 outputs are identical to the original except for a few spurious typographic and punctuation changes. Because of that, Nematus simplifications are in average four tokens shorter than both complex originals and Moses simplifications.

A strong limitation of both models is their inability to address lexical complexity, performing very few lexical replacements. Most of the sentences that are lexically simplified have only one word replaced by another that does not preserve its original meaning. Take, for example, the word *clears* in the sentence "*It clears the way for troops on the ground with its huge bullets*", which was replaced by *gathers* by Nematus, and the word *agribusiness*, which was replaced by *offering* by Moses in sentence "*Older brother Nate has taken college courses on livestock raising and agribusiness*". Some of these issues become more evident in the human evaluation we performed comparing both end-to-end systems to our proposed approach (§5.2).

4 Simplification via Sequence Labeling

Our approach to TS differs from translation-based models by explicitly predicting a set of operations to be applied at different positions in a complex sentence. Concretely, we tackle simplification as a sequence labeling problem, predicting operations at the token level and applying them downstream. As there are no high-quality and large-scale resources from which such operation sequences could be learned, we first generate training data as explained below.[6]

4.1 Generating Training Data

Given 1-to-1 sentence pairs, our method for data generation identifies deletions, additions, substitutions, rewrites (replacing or adding non-content words), and reorderings performed between sentences pairs.

Automatic operation annotation. The annotation process uses the following set of operation labels: DELETE (D), REPLACE (R), and MOVE (M) in the original (source) sentence; ADD (A) in the simplified sentence; and REWRITE (RW) in both.[7]

We first generate word alignments between the original and simplified sentences using the aligner by Sultan et al. (2014). Based on these alignments, we perform a word-level annotation for labels DELETE and REPLACE. Our heuristics are that if two words are aligned and are not an exact match, then the corresponding label is REPLACE. If a word in the original sentence is not aligned, it

[6]For the experiments with the proposed TS approach, only 1-to-1 alignments are suitable. It is indeed not realistic to expect that complex operations that involve significant structural changes (e.g., splitting or joining sentences) could be modeled using sequence labeling approaches. For such complex operations, we believe explicitly representing the sentences' syntactic structures and learning abstract syntactic transformation rules (e.g. as in Woodsend and Lapata (2011) or Feblowitz and Kauchak (2013)) would be more advisable. However, we note that, as previously shown, translation-based end-to-end approaches also fail to learn such complex operations.

[7]Target-side annotations serve for analysis; they are ignored in our experiments as they are unavailable at test time.

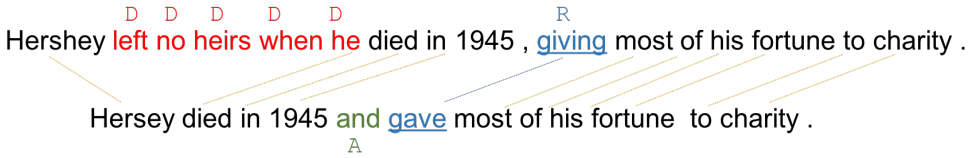

Figure 1: Example of automatic labeling based on word alignments between an original (top) and a simplified (bottom) sentence in the Newsela corpus. Unaligned words on the original side receive label 'D' (DELETE), while words that are aligned to a different form receive 'R' (REPLACE). Aligned words without an explicit label receive a 'C' label (COPY). Sentences are from the Newsela Article Corpus.

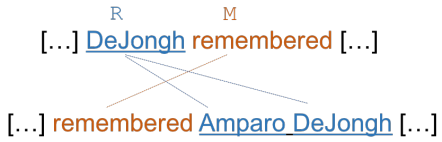

Figure 2: Example of automatic annotation for label MOVE ('M'). Sentences are from the Newsela Article Corpus.

must be a DELETE, and if a word in the simplified sentence is not aligned, it is an ADD. In any other case, the word receives label C (COPY) or O (not part of a simplification operation) in the original or simplified sentence, respectively. For details, see Algorithm 1 in the supplementary material. Figure 1 presents an example for our automatic labeling approach. We consider REWRITE labels as special cases of REPLACE where the words involved are isolated (not in a group of same operation labels) and belong to a list of non-content words.

Finally, we label reorderings (MOVE) by determining if the relative index of a word (considering preceding or following deletions and additions) in the original sentence changes in the simplified one (Algorithm 2). See Figure 2 for an example. Words or phrases that are kept, replaced or rewritten, may be subject to reorderings, such that a token may have more than one label (e.g. REPLACE and MOVE). For that, we extend the set of operations by the compound operations REPLACE+MOVE (RM) and REWRITE+MOVE (RWM).

Evaluation of automatic labels. To test our algorithms, we compare their output to manual annotations for 100 sentences from level pair 0-1 of the Newsela corpus. The manual annotations were performed by four proficient English speakers. For 30 of those sentences, we calculated the pairwise inter-annotator agreement between annotators, yielding an average kappa value of 0.57. We obtain an accuracy of 0.92 for all labels, and a micro-averaged F_1 score of 0.70 for all positive labels (i.e. excluding 'C' and 'O'). Table 3 presents details on the performance of our annotation algorithms over the identified operations. Of the positive labels, the algorithms annotate most accurately additions and deletions. According to the confusion matrix in Table 4, the relatively low ability of capturing replacements is due to labeling them as deletions. This is mainly caused by word miss-alignments and by parser errors that our heuristics cannot recover from. The same logic applies for labels REPLACE+MOVE and REWRITE+MOVE. We are also able to capture most MOVEments (high recall), but our reordering heuristic still requires improvement.

Label	Prec.	Rec.	F_1	Support
A	0.66	0.92	0.77	261
D	0.76	0.90	0.82	371
M	0.17	0.92	0.28	24
R	0.70	0.39	0.50	71
RM	0.22	0.33	0.27	12
RW	0.24	0.07	0.11	57
RWM	0.00	0.00	0.00	6
C	0.99	0.94	0.96	1932
O	0.99	0.95	0.97	2112
avg / total	0.92	0.92	0.92	4846

Table 3: Per-label performance of automatic annotation of operations.

We refer to these automatically generated labels as **silver labels**. As we describe in the next sections, the corpus annotated with these labels will be used to train our sequence labeling approach, eliminating the need for costly human-annotated data (i.e. gold labels). As a second way of evaluating the quality of our automatic labeling, we use these silver labels in a semi-oracle trial where we apply the actual simplification operations as given in the annotated corpus. In other words, we simply take the automatic labels as true and use the

	Automatically Annotated								
	A	D	M	R	RM	RW	RWM	C	O
A	240	0	0	0	0	2	0	0	19
D	15	333	8	4	5	1	1	4	0
M	0	1	22	0	0	0	0	1	0
R	0	33	0	28	6	0	0	4	0
RM	0	8	0	0	4	0	0	0	0
RW	3	31	4	7	2	4	0	6	0
RWM	0	6	0	0	0	0	0	0	0
C	0	24	98	1	1	1	0	1807	0
O	105	0	0	0	0	9	0	0	1998

Table 4: Confusion matrix of true (rows) and automatically annotated (columns) operations on the manually annotated data.

alignments between original and simplified words to apply the actual operations. This is what we refer to as **silver operations** in Table 1. Using the automatic labeling would lead to much more accurate and less conservative simplifications than all translation-based approaches: it achieves the highest SARI and BLEU scores, and the lowest rate of copied input sentences among all systems tested using the 1-to-1 alignments. Therefore, the challenges now are (i) to predict such labels (§5.1), and (ii) to devise high-performing TS modules to apply simplification operations for each type of label (§4.2).

4.2 Application of Operations

For our experiments (§5), we consider two of the operations that our algorithms can identify with high precision: DELETE and REPLACE.[8] Applying deletions is straightforward and amounts to simply omitting the respective token when generating the hypothesis sentence. For the REPLACE operation, we use the supervised Lexical Simplification approach of Paetzold and Specia (2017). Their simplifier generates candidate substitutions for target words using parallel complex-to-simple corpora and retrofitted context-aware word embedding models, selects the ones that fit the context of the target word through the unsupervised boundary ranking approach, then ranks candidates using a supervised neural ranking model trained over manually annotated simplifications. It also performs a final confidence check step: the target is only replaced by the highest ranking candidate if the trigram probability of two words preceding the target is higher for the candidate.

5 Experiments

Based on the automatic annotation procedure outlined above, we generate sequence annotations of 1-to-1 simplification operations in the Newsela corpus. On this data, we explore the questions (i) whether we can predict simplification operations to be performed on unseen data, and (ii) to what degree the prediction of these operations allows us to generate good simplifications.

5.1 Prediction of Simplification Operations

To predict simplification operations for each input word, we train a bidirectional recurrent neural network, with an initial embedding layer of size 300 and two hidden LSTM (Long-Short Term Memory) layers of size 100. The training is done using Keras (Chollet, 2015), with a batch size of 64, categorical cross-entropy loss and a dropout rate of 0.2 after the hidden layers. We optimize the model with Adagrad (Duchi et al., 2011). We monitor the tagging accuracy on held-out development data and employ early stopping when the development loss increases. We repeat this process ten times with random initializations and select the best model based on development set accuracy.

Table 5 shows that the LSTM model does not predict the silver labels very well. In particular, the model is relatively conservative with respect to the prediction of simplification operations, and tends to overpredict the majority class (i.e., to copy a token).[9] DELETE is the operation that our model predicts best. Table 6 shows the relative confusion of predicted operations versus the silver labels, and confirms that the main error type of our system is to keep a token rather than performing some simplification operation on it. We also see a tendency for other operations to be predicted as deletions.

The results in the lower part of Table 1 ("Predicted operations (1-to-1)"), however, show that even though the operation predictions are far from the silver labels, our system is able to generate simple output by only applying the DELETE and

[8] We focus on this subset of operations since we currently lack good models to apply to the remaining operations. ADD, for example, would presume access to an external resource such as a knowledge base that would serve as a basis for inferring added content (which is oftentimes background information, for example an explanation that a certain person has a certain function). The results we obtain can thus be viewed as a lower bound on the simplification quality that can be expected from a model that integrates other operations.

[9] By weighting the loss function by the ground truth class support at each timestep, we were able to alleviate the effect of a predominant majority class to some degree.

Label	Prec.	Rec.	F_1	Support
D	.30	.49	.37	58,692
M	.21	.16	.18	29,719
R	.13	.34	.19	7,208
RM	.00	.00	.00	2,817
RW	.14	.07	.10	646
RWM	.00	.00	.00	141
C	.68	.51	.58	154,481
avg / total	.51	.45	.47	253,704

Table 5: Per-label performance of automatic operation prediction with the LSTM model.

	Predicted						
	D	M	R	RM	RW	RWM	C
D	.49	.06	.07	.00	.00	.00	.38
M	.41	.16	.05	.00	.00	.00	.38
R	.23	.05	.34	.00	.00	.00	.38
RM	.32	.09	.21	.00	.00	.00	.38
RW	.38	.00	.00	.00	.07	.00	.54
RWM	.62	.03	.00	.00	.04	.00	.32
C	.33	.09	.06	.00	.00	.00	.51

Table 6: Confusion matrix of true (rows) and predicted (columns) operations on the test data.

REPLACE operations. In particular, our method achieves a better SARI score than all the baseline systems on the 1-to-1 alignments. As we consider the extrinsic evaluation of the final TS results to be more indicative of the quality of our model than its intrinsic evaluation in the sequence labeling task, we view this as a positive result.

5.2 Human Evaluation

We finally conduct a human evaluation of 100 simplifications produced by five simplifiers:

- The experts' **Reference** simplification.
- The **Moses** simplifier (1-to-1).
- The **Nematus** simplifier (1-to-1).
- The **NTS** simplifier (1-to-1).
- Our **Sequence Labeling (SL)** simplifier.

Human evaluators (four NLP experts) are given the original sentence and the simplification in each of the above versions, and are asked to judge each of them with respect to their grammaticality (G), meaning preservation (M) and simplicity (S), using a Likert scale between 1 (worst) and 5 (best) for each aspect. We define "simplicity" as the extent to which the sentence was simpler than the original and thus easier to understand. A control set of 20 sentences is evaluated by all annotators in order to compute inter-annotator agreement.

	G	M	S
Reference	5.00±0.0	4.45±0.9	2.70±1.3
SL	4.16±1.0	3.91±1.1	1.66±0.9
Nematus	4.49±0.9	3.99±1.2	1.46±0.9
Moses	4.98±0.2	4.99±0.1	1.14±0.4
NTS	4.75±0.6	4.08±1.26	1.53±1.0
Fleiss' Kappa	0.372	0.457	0.342

Table 7: Average scores and standard deviation for grammaticality (G), meaning preservation (M) and simplicity (S) for the systems evaluated. The last row shows the inter-annotator agreement scores in terms of Fleiss' Kappa.

Table 7 illustrates the average scores and standard deviations obtained by each system according to each criterion. As expected, the Moses simplifier obtains the highest grammaticality and meaning preservation scores, but the lowest simplicity scores, given that it tends to merely reproduce the input. Although Nematus and NTS manage to obtain slightly higher simplification scores, they still average very close to the lower end of the simplicity scale. Our SL approach, in turn, shows significantly higher simplicity scores than the other systems (according to a t-test with $p < 0.05$). Its less conservative edits, however, may in some cases come at the cost of lower scores for grammaticality and meaning preservation. The last row in Table 7 shows the values of inter-annotator agreement in terms of Fleiss' Kappa for each evaluation aspect. Table 8 exemplifies some of the sentences for which our system was rated better and worse than the baselines. It is important to mention that, although the first two reference simplifications in Table 8 feature only minor punctuation changes, only 2,538 references (0.8%) in the dataset are of this type.

6 Conclusions and Further Work

We presented a novel approach to sentence simplification that uses automatically labeled training data from a large simplification corpus. Based on this annotated corpus, we devise a sequence labeling approach to text simplification that predicts simplification operations for individual words in the original sentence. Specific modules are then triggered to deal with each predicted operation.

	SL better than other Moses, Nematus and NTS
O	Kyarra Garrett has learned how to take blood pressure and perform CPR – and she is not even out of high school yet.
R	Kyarra Garrett has learned how to take blood pressure and perform CPR, and she is not even out of high school yet.
M	*Kyarra Garrett has learned how to take blood pressure and perform CPR – and she is not even out of high school yet.*
N	*UNK Garrett loves out to take blood pressure and perform, and she is not even out of high school yet.*
T	*Chance Garrett has learned how to take blood pressure.*
L	*Kyarra Garrett has learned how to take blood pressure and perform CPR.*
O	in her mind she stops at particular locations to pick up the correct cookie crumbs.
R	in her mind, she stops at particular locations to pick up the correct cookie crumbs.
M	*in her mind she stops at particular locations to pick up the correct cookie crumbs.*
N	*she stops at particular locations to pick up the correct cookie UNK.*
T	*in her mind she stops at particular locations to pick up the correct cookie momentum.*
L	*in her mind she stops at particular areas to pick up cookie crumbs.*
	SL worse than Moses, Nematus or NTS
O	despite the limitations, Palestinian cooking is not without its fans.
R	despite the limitations, Palestinian cooking has its fans.
M	*despite the limitations, Palestinian cooking is not without its fans.*
N	*Palestinian cooking is not without its fans.*
T	*even Palestinian cooking is not without its fans.*
L	*despite the limitations, Palestinian cooking is not without its fans.*
O	"we always thought there has to be a more efficient way of doing this," Zach Fiene said.
R	he said he always thought there had to be a better way of doing it.
M	*"we always thought there has to be a more efficient way of doing this," Zach Fiene said.*
N	*"we always thought there has to be a more efficient way of doing this," said Zach Ghani, who is the 18-year-old said.*
T	*Zach Fiene said there has to be a more efficient way of doing this.*
L	*"we always thought there has to be more efficient way doing this said.*

Table 8: Example including original (O) and reference (R) sentences from the Newsela Article Corpus, and outputs generated by Moses (M), Nematus (N), NTS (T) and our sequence labeling approach (**L**).

The experiments reported here cover only deletions and lexical substitutions as operations.

Our approach has several theoretical advantages over end-to-end translation models, including easier interpretability of the types of simplification learned, as well as the possibility for late decoding for adaptive simplification. In practical terms, we showed that our system outperforms translation-based approaches on a number of metrics and overcomes the problems of excessive repetition of the original content.

According to human evaluation, our system achieves higher simplicity scores than the baseline systems, although this comes at the cost of slightly lower meaning preservation and grammaticality. We hypothesize that some of the problematic cases stem from not realizing the addition operation. In general, our approach will likely profit from good models for the remaining operations, especially those that can also operate on spans of several tokens, making research on such models a natural direction for further work.

Acknowledgements

This work was partly supported by the EC project SIMPATICO (H2020-EURO-6-2015, grant number 692819).

References

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

Regina Barzilay and Noemie Elhadad. 2003. Sentence alignment for monolingual comparable corpora. In *Proceedings of EMNLP*, pages 25–32.

Joachim Bingel and Anders Søgaard. 2016. Text Simplification as Tree Labeling. In *Proceedings of ACL*, pages 337–343.

Stefan Bott and Horacio Saggion. 2011. An unsupervised alignment algorithm for text simplification corpus construction. In *Proceedings of MTTG*, pages 20–26.

François Chollet. 2015. Keras. `https://github.com/fchollet/keras`.

William Coster and David Kauchak. 2011a. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*, pages 1–9.

William Coster and David Kauchak. 2011b. Simple English Wikipedia: a new text simplification task. In *Proceedings of ACL*, pages 665–669.

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research*, 12:2121–2159.

Dan Feblowitz and David Kauchak. 2013. Sentence simplification as tree transduction. In *Proceedings of the PITR Workshop*, pages 1–10.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of ACL - Demonstration Session*, pages 177–180.

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

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

Gustavo H. Paetzold and Lucia Specia. 2016a. Semeval 2016 task 11: Complex word identification. In *Proceedings of the 10th SemEval*, pages 560–569.

Gustavo Henrique Paetzold and Lucia Specia. 2016b. Vicinity-driven paragraph and sentence alignment for comparable corpora. *CoRR*, abs/1612.04113.

Gustavo Henrique Paetzold and Lucia Specia. 2017. Lexical simplification with neural ranking. *Proceedings of EACL*, pages 34–40.

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

Rico Sennrich, Orhan Firat, Kyunghyun Cho, Alexandra Birch, Barry Haddow, Julian Hitschler, Marcin Junczys-Dowmunt, Samuel Läubli, Antonio Valerio Miceli Barone, Jozef Mokry, and Maria Nadejde. 2017. Nematus: a toolkit for neural machine translation. In *Proceedings of the Software Demonstrations of EACL*, pages 65–68, Valencia, Spain. ACL.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for wmt 16. In *Proceedings of WMT*, pages 371–376.

Jason R Smith, Chris Quirk, and Kristina Toutanova. 2010. Extracting parallel sentences from comparable corpora using document level alignment. In *Proceedings of NAACL*, pages 403–411.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of AMTA*, pages 223–231.

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

Md Sultan, Steven Bethard, and Tamara Sumner. 2014. Back to basics for monolingual alignment: Exploiting word similarity and contextual evidence. *TACL*, 2:219–230.

Kristian Woodsend and Mirella Lapata. 2011. Learning to simplify sentences with quasi-synchronous grammar and integer programming. In *Proceedings of EMNLP*, pages 409–420.

Sander Wubben, Antal van den Bosch, and Emiel Krahmer. 2012. Sentence simplification by monolingual machine translation. In *Proceedings of ACL*, pages 1015–1024.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *TACL*, 3:283–297.

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

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. In *Proceedings of EMNLP*, pages 595–605.

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

Zhemin Zhu, Delphine Bernhard, and Iryna Gurevych. 2010. A monolingual tree-based translation model for sentence simplification. In *Proceedings of COLING*, pages 1353–1361.

A Supplementary Material

Algorithm 1: Initial word-level annotation

Input: O: list with tokens of the original sentence, S: list with tokens of the simplified sentence, A: list with word alignments.
Output: SLO: simplification labels for each token in O, SLS: simplification labels for each token in S.

```
// labeling tokens in the original sentence
for i ← 1 to len(O) do
    // get the indexes of the tokens in S to which the ith token in O is aligned to
    IS ← FindAlignments(A, i, 's')
    if len(IS) > 0 then                          // it is aligned
        if len(IS) = 1 and O_i = S_{IS_0} then
            SLO_i ← 'C'                          // keep
        else                                     // not an exact match
            SLO_i ← 'R'                          // replace
        end
    else                                         // not aligned
        SLO_i ← 'D'                              // delete
    end
end
// labeling tokens in the simplified sentence
for j ← 1 to len(S) + 1 do
    // get the indexes of the tokens in O to which the jth token in S is aligned to
    IO ← FindAlignments(A, j, 'o')
    if len(IO) > 0 then                          // it is aligned
        SLS_j ← 'O'
        if len(IO) > 1 then
            // the current token in S replaces a phrase in O
            foreach k ∈ IO do
                SLO_k ← 'R'
            end
        end
    else
        SLS_j ← 'A'                              // add
    end
end
```

Algorithm 2: Annotation of reorderings

Input: SLO: simplification labels for each token in original sentence, SLS: simplification labels for each token in simplified sentence, A: list with word alignments.
Output: SLO modified.

```
shift_left ← 0
for i ← 0 to len(SLO) do
    if SLO_i ∈ ['D', 'R'] then
        shift_left ← shift_left + 1
    else
        IS ← FindAlignments(A, i, 's')
        if len(IS) > 0 then
            k ← IS_0                             // index of the aligned token in the simplified sentence
        else
            k ← i                                // index of the token in the original sentence
        end
        shift_right ← 0
        for j ← 0 to k do
            if SLS_j ∈ ['AC', 'RW'] then
                shift_right ← shift_right + 1
            end
        end
        if i − shift_left + shift_right ≠ k then
            switch SLS_i do
                case 'C' do SLS_i ← 'M'
                case 'R' do SLS_i ← 'RM'
                case 'RW' do SLS_i ← 'RWM'
            end
        end
    end
end
```