On the Helpfulness of Document Context to Sentence Simplification

Renliang Sun, Zhe Lin, Xiaojun Wan
Wangxuan Institute of Computer Technology, Peking University
Center for Data Science, Peking University
The MOE Key Laboratory of Computational Linguistics, Peking University
sunrenliangpku@gmail.com
{linzhe, wanxiaojun}@pku.edu.cn

Abstract

Most of the research on text simplification is limited to sentence level nowadays. In this paper, we are the first to investigate the helpfulness of document context on sentence simplification and apply it to the sequence-to-sequence model. We firstly construct a sentence simplification dataset in which the contexts for the original sentence are provided by Wikipedia corpus. The new dataset contains approximately 116K sentence pairs with context. We then propose a new model that makes full use of the context information. Our model uses neural networks to learn the different effects of the preceding sentences and the following sentences on the current sentence and applies them to the improved transformer model. Evaluated on the newly constructed dataset, our model achieves 36.52 on SARI value, which outperforms the best performing model in the baselines by 2.46 (7.22%), indicating that context indeed helps improve sentence simplification. In the ablation experiment, we show that using either the preceding sentences or the following sentences as context can significantly improve simplification.

1 Introduction

Text simplification is a hot issue in the field of natural language generation (NLG). It is also one of the critical needs of society (Woodsend and Lapata, 2011). Text simplification aims to adapt a complex text into a more readable version with the same meaning (Sulem et al., 2018b), which will benefit young children (Kajiwara et al., 2013) and non-native English speakers (Paetzold, 2015; Paetzold and Specia, 2016). It includes many ways to deal with the input text, such as deletion, reordering, paraphrase, and sentence separation (Saggion, 2017). Besides, text simplification is closely related to many natural language processing (NLP) tasks, such as machine translation (Štajner and Popović, 2016; Hasler et al., 2017), paraphrase generation (Pavlick and Callison-Burch, 2016; Cao et al., 2017; Zhao et al., 2018) and text summarization (Li et al., 2017; Ma and Sun, 2017; Jin et al., 2020).

In recent years, many researchers have proposed various models to improve the performance of text simplification. However, few people have explored the impact of document context (i.e., the preceding and following sentences of the original sentence to be simplified in a document) on text simplification, let alone establish a large-scale dataset containing context. Many examples like the one in Table 1 arouse our interest in exploring the influence of context. In the simplified sentence, "played and sang" is the simplification of "in her performances" in the original sentence. The phrase "sing as well as play" in the context may provide additional information to help the simplification. The phrase "at the bar" is retained because of the presence of "at the Midtown Bar" in the context. Correspondingly, there is no mention of information related to "fan base" in the context, so adjectives such as "loyal", "small" are deleted.

In this paper, we are committed to investigating the influence of document context on text simplification and proposing a neural model to improve simplification using context. Using the Wikipedia datasets (Coster and Kauchak, 2011; Kauchak, 2013), we first construct a dataset in which the document context

Proceedings of the 28th International Conference on Computational Linguistics, pages 1411–1423
Barcelona, Spain (Online), December 8-13, 2020

Context	*To fund her private lessons , Simone performed* ***at the Midtown Bar*** *& Grill on Pacific Avenue in Atlantic City , whose owner insisted that she* ***sing as well as play*** *the piano .*
Original	*Simone 's mixture of jazz , blues , and classical music* ***in her performances at the bar*** *earned her a small , but loyal , fan base .*
Simplified	*Simone* ***played and sang*** *a mixture of jazz , blues and classical music* ***at the bar*** *. She began to get fans .*

Table 1: An example of context affecting simplification. We use **Bold** to highlight the keywords.

of the original sentence in each sentence pair is provided[1]. This dataset is built automatically, and the training set has more than 150K sentence pairs. Then, we propose a new model named simplification using context (SUC) by improving the transformer model. We use two multi-head self-attention modules to learn the representations of the context. We also use neural networks to learn the different weights and multiply them with the self-attention layer's output. In the end, we conduct experiments to explore the different effects of the preceding sentences and the following sentences on the current sentence.

The experiment results verify that our model has achieved remarkable results. Compared with the original transformer model, our model improves the SARI value by 4.05 points. The ablation experiments show that the use of sentence pairs with context information can significantly improve the SARI points.

There are the following main contributions of our work:

(1) We are the first to investigate the influence of document context on sentence simplification and build a large dataset for training and testing.

(2) We propose and train a new model named simplification using context (SUC), which makes full use of context information. SUC outperforms the baselines on both automatic evaluation and human evaluation.

(3) We use ablation experiments to illustrate the different effects of the preceding sentences and the following sentences on the current sentences to be simplified.

2 Related works

Text simplification has developed rapidly in the past decade. Wubben et al. (2012) proposed the PBMT-R model while Narayan and Gardent (2014) put forward the Hybrid model. Both models were based on statistic machine learning. Using a small number of manual simplifications and a large number of paraphrases, Xu et al. (2016) adapted the method of the statistical machine translation. Nisioi et al. (2017) were the first to apply the sequence-to-sequence model to automatic text simplification, using the framework of neural machine translation and making specific improvements. Zhang and Lapata (2017) proposed the DRESS model which rewards the simple, fluent output sentences with proper meaning. Vu et al. (2018) used memory-augmented neural networks to adapt the existing architecture and Guo et al. (2018) used multi-task learning to improve the entailment and paraphrase capabilities.

Recently, Kriz et al. (2019) used two techniques to solve the problem that the model tends to copy words directly, resulting in a long and complicated output sentence. Nishihara et al. (2019) proposed a method to simplify the original sentences to different level sentences. Different from most sequence-to-sequence models, Dong et al. (2019) proposed a neural programmer-interpreter approach to predict explicit edit operations directly. Jiang et al. (2020) proposed the neural CRF model to get better sentence alignment.

However, up to now, most of the research on text simplification is limited to sentence level, ignoring the influence of document context on sentence simplification. Pitler and Nenkova (2008) are the first to empirically demonstrate that discourse relations are closely related to the perceived quality of the text. So far, Zhong et al. (2020) are the first and the only ones to focus on discourse level factors of text simplification. Their results have shown that using discourse level factors is useful for predicting sentence deletion. Nevertheless, different from our research focusing on sentence simplification, this work mainly

[1]The dataset we used in the experiment is available at https://github.com/RLSNLP/Document-Context-to-Sentence-Simplification

analyzes and predicts the sentence deletion in document simplification. In addition to deletion operation, sentence simplification also includes reservation, separation, synonym replacement, and other operations (Xu et al., 2016). Whether the preceding sentences and the following sentences have different effects on the original sentence has not been taken into account.

Even in a similar field of machine translation, most of the research focuses on the preceding sentences. Based on the transformer model, Zhang et al. (2018) used a new encoder to represent the context and proposed a new model. Werlen et al. (2018) proposed a hierarchical attention model that captures the context in a structured and dynamic way. Both of the models that have received widespread attention only focuses on the preceding sentences. In addition to the preceding sentences, the following sentences also contain the information of keeping, paraphrasing, and deleting the words in the original sentence in text simplification. To the best of our knowledge, we are the first to study the effects of the preceding sentences and the following sentences and apply them to a sequence-to-sequence model in the field of text simplification.

3 Our Model

The SUC model consists of four parts: the transformer model, context information module, pointer-generator network, and coverage mechanism. Among them, the context information module is an original module proposed by us to get the context representation and apply it to the transformer model.

3.1 The Transformer Model

The transformer model is based on attention mechanism, and the model has a straightforward structure (Vaswani et al., 2017). The original transformer model has an encoder and a decoder, and both of them use multi-head attention mechanism and feed-forward networks. It turns a set of queries into a matrix Q, and turns the keys and values into matrix K and V respectively. Define the dimension of queries and keys as d_k and values as d_v, and the output can be computed as:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{1}$$

A multi-head attention is defined as follows:

$$\begin{aligned} MultiHead(Q, K, V) &= Concat(head_1, ..., head_h)W^O \\ where\ head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2}$$

Where $W_i^Q \in R^{d_{model} \times d_k}$, $W_i^K \in R^{d_{model} \times d_k}$, $W_i^V \in R^{d_{model} \times d_v}$ and $W^O \in R^{d_{model} \times hd_v}$, and $d_k = d_v = d_{model}/h$.

3.2 Context Information Module

3.2.1 Representation of Context Information

First, we will give an overview of the module of computing context information representation, as shown in Figure 1. We use two additional encoders to calculate the representations of the preceding sentences and following sentences, respectively. The input text first passes through the embedding layer and the position encoding layer. Define $X \in R^{D \times L}$ as the representation of input text after embedding and position encoding where D is embedding dimension and L is the length of input text.

The additional encoder consists of N identical layers, each of which contains two components. N is equal to the number of layers of the encoder and decoder. The first component is a multi-head attention, which is the same as the one in the transformer model. The input matrix Q, K, and V of multi-head attention are all matrix X. The formula is defined as:

$$Attn = MultiHead(X, X, X) \tag{3}$$

The second component is a fully connected network. The network consists of two linear transformation layers and a GELU activation layer, the same as the one in the transformer model. Thus the attention result of the input text is obtained.

$$R = FFN(Attn) \tag{4}$$

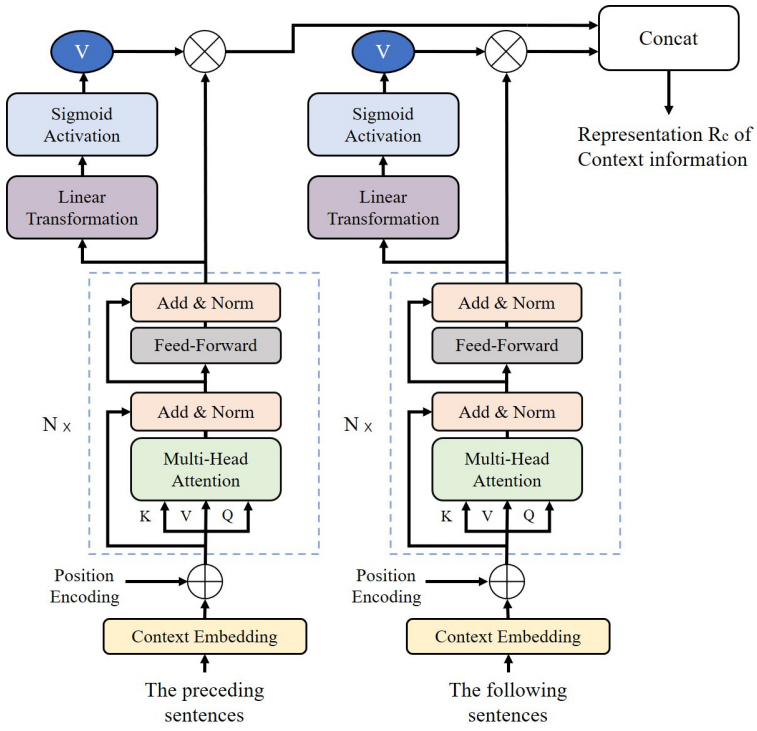

Figure 1: An overview of how to get the representation of the context information. The inputs are the preceding sentences and the following sentences of the current sentence. The output is the representation R_C of the context information.

It is worth noting that in order to prevent overfitting, the dropout mechanism and the LayerNorm is added at the end of each component, which is defined as:

$$O = LayerNorm(I + Dropout(O)) \tag{5}$$

The symbols I and O represent the input and output of the component respectively.

Given the input matrix X, the output $R_N \in R^{d_f \times L}$ of the additional encoder is obtained where d_f represents the dimension of the inner-layer. We design a new neural network to calculate the weight V corresponding to R_N. The network consists of a linear transformation layer and a sigmoid activation layer. The linear transformation layer converts R_N to weights V with dimension 1. The sigmoid activation layer converts V to a value between 0 and 1. When the input of the sigmoid function is near 0, the derivative is larger. When the input approaches positive infinity or negative infinity, the output approaches 1 or 0 respectively. The calculation process of V can be defined as:

$$\begin{aligned} V &= wR_N + b \\ V &= Sigmoid(V) \end{aligned} \tag{6}$$

Then we multiply V by R_N to get the weighted output of the additional encoder. There are two additional encoders in our model, which are used to obtain the representations for the preceding sentences and the following sentences, respectively. We combine two weighted outputs to get R_C, which is defined as:

$$R_C = Concat(V_1 R_{N1}, V_2 R_{N2}) \tag{7}$$

R_C is the representation of the context information. V_1 is different from V_2, indicating that the weights of the preceding sentences and the following sentences are different.

3.2.2 Incorporation of Context Information

We also give an overview of how to use the context information, as shown in Figure 2. We apply the representation R_C of context information to encoder and decoder at the same time. The encoder and the decoder contain N identical improved layers. We also use the dropout mechanism and LayerNorm, as we use in the additional encoder.

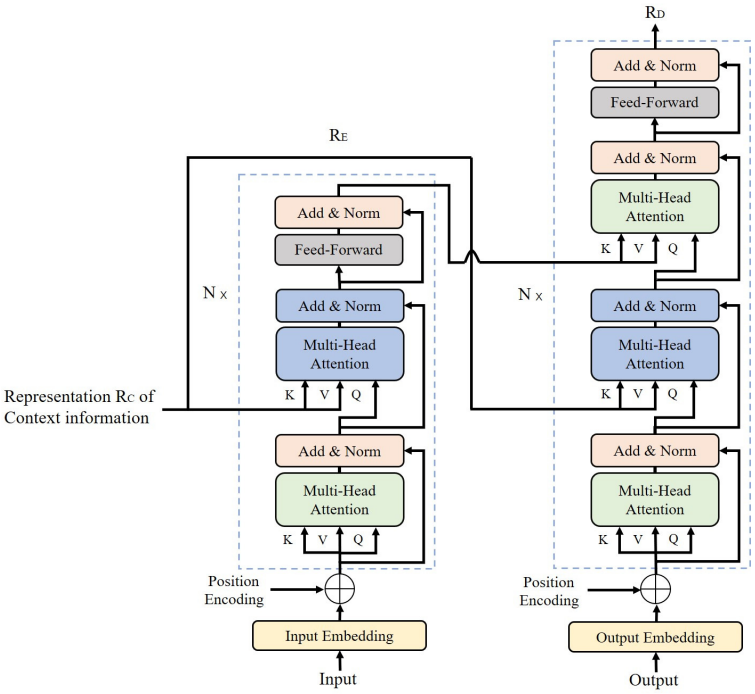

Figure 2: An overview of how to use the context information. The blue modules are the ones we add to process the representation of context information.

The improved encoder layer consists of three components, including two multi-head attentions and a neural network. The first component is a multi-head attention. Define X_{origin} as the representation of the original sentence after embedding and position encoding. The input matrix Q, K, and V of the first multi-head attention are all matrix X_{origin}, which is defined as:

$$Attn = MultiHead(X_{origin}, X_{origin}, X_{origin}) \tag{8}$$

The second component is also a multi-head attention. We input the representation R_C of the context into this multi-head attention as matrices K and V. Matrix Q is the output $Attn$ of the previous multi-head attention. The formula can be defined as:

$$Attn' = MultiHead(Attn, R_C, R_C) \tag{9}$$

The last component is a neural network, which is the same as the one in the additional encoder. Define R_E as the output of encoder, and the formula can be expressed as:

$$R_E = FFN(Attn') \tag{10}$$

The improved decoder layer consists of four components, including three multi-head attentions and a neural network. Following the original transformer model, we offset the word embedding of the simple sentence by one position. Define that X_{simple} is the representation of the simple sentence after embedding and position encoding. The input matrix Q, K, and V of the first multi-head attention are all matrix X_{simple}, which is defined as:

$$Attn = MultiHead(X_{simple}, X_{simple}, X_{simple}) \tag{11}$$

The second component is the same as the second component in encoder layer. It also takes the representation R_C of the context and output of the first component as input, which is defined as:

$$Attn' = MultiHead(Attn, R_C, R_C) \tag{12}$$

The third component is a multi-head attention but we take the output R_E of the encoder as matrices K and V. The matrix Q is the output $Attn'$ of the previous multi-head attention. The formula can be defined as:

$$Attn'' = MultiHead(Attn', R_E, R_E) \tag{13}$$

The last component is a neural network, which is the same as the one in encoder. Define R_D as the output of the decoder, and the formula can be expressed as:

$$R_D = FFN(Attn'') \tag{14}$$

3.3 Pointer-Generator Network and Coverage Mechanism

The pointer-generator network copies words from the original sentence to solve the problem of out-of-vocabulary (See et al., 2017). Following the implementation[2], the pointer-generator network used in our model contains a multi-head attention. R_D is taken as matrix Q and R_E is taken as matrices K and V, which is defined as:

$$A = MultiHead(R_D, R_E, R_E) \tag{15}$$

The generation probability P_{gen} can be calculated as:

$$P_{gen} = Sigmoid(W_1 A + W_2 R_D + W_3 X_{simple} + b) \tag{16}$$

The vectors W_1, W_2, W_3 and the scalar b are all learnable parameters. The final probability distribution can be defined as:

$$P(W) = P_{gen} P_{vocab} + (1 - P_{gen}) D \tag{17}$$

Where D can be obtained from multi-head attention and P_{vocab} is the vocabulary.

The coverage model focuses on solving the problem of generating text repeatedly in sequence-to-sequence model (Tu et al., 2016). The general coverage model is given by:

$$C_i = g_{update}(C_{i-1}, \alpha_i, \Phi(h), \Psi) \tag{18}$$

C_i is the coverage vector which summarizes the previous attentions at time step i to help adjust future attention. g_{update} updates C_i after new attention α_i when decoding at time step i. $\Phi(h)$ is a word-specific feature and Ψ are different auxiliary inputs.

[2]The code is available at `https://github.com/lipiji/TranSummar`.

4 Experiments

4.1 Dataset

Since the contexts are not provided in commonly used datasets such as Wikismall (Zhu et al., 2010) and Newsela (Xu et al., 2015), we need to build appropriate datasets first. With the help of the Wikipedia datasets (Coster and Kauchak, 2011; Kauchak, 2013), we successfully construct a dataset for our research[3]. The Wikipedia datasets contain about 167K aligned sentence pairs. We extract the context sentences of each original sentence from the document-aligned data. For those sentences without the preceding sentences or the following sentences, we choose to retain them to train the sentence-level modules of our model. In the training set, there are around 110K aligned sentence pairs with context information and around 41K aligned sentence pairs without context information. From the remaining sentence pairs with context information, we use 5K as the validation set and 1K as the test set. There is no repetition among the sentences in the test, validation and training sets.

Previous study on machine translation has shown that using too much context information will not only not improve the results, but increase the computational complexity (Tu et al., 2018). Following Zhang et al. (2018), we take two preceding sentences and two following sentences of the current sentence as context information. If there is only one preceding sentence or following sentence, we choose to keep it.

4.2 Evaluation Metrics

We use SARI (Xu et al., 2016) and FKGL (Kincaid et al., 1975) and BLEU (Papineni et al., 2002) as automatic evaluation metrics in our work.

The SARI metric may be the most important criteria to measure the result of text simplification.[4] The SARI metric compares the simplified sentence with the original one and the reference one at the same time. The SARI value comes from three parts: adding words, deleting words properly and keeping words properly. The values of the three parts are also reported in our results.

The BLEU metric is commonly used previously and used to measure the similarity of output to a reference sentence (Zhao et al., 2020), so we decide to use this metric.[5] It is worth noting, however, that the BLEU metric has been found often negatively correlates with simplicity recently (Sulem et al., 2018a).

Following Dong et al. (2019), we choose to use the FKGL method to measure the readability of the output sentences.[6] The lower the FKGL value, the simpler the output sentences are.

In this paper, we regard SARI as the most important criterion to judge the effect of simplification. We also employ human judges to conduct more reliable evaluation for this task.

4.3 Training Details

Our model is based on the transformer model (Vaswani et al., 2017). The addition encoder, encoder and decoder in our model have 4 layers with 4 multi-heads. We set the size of the vocabulary to 45800 and other uncommon words in the training set are replaced with the out-of-vocabulary token UNK. When predicting, we follow the method proposed by Jean et al. (2015) to replace UNK. We use the Adagrad optimizer (Duchi et al., 2011) to train our model and we train 50 epochs. We set the learning rate to 0.1 and the training batch size to 16. Following BERT (Devlin et al., 2019), we replace the ReLU activation function with a GELU activation function (Hendrycks and Gimpel, 2016) that performs better on transformer. When training, we firstly use the sentence pairs with context information to train the whole model, then we fix the parameters of the document-level module. Next, we use the sentence pairs without context information to train the sentence-level module.

[3]We originally planned to make a dataset based on Newsela at the same time, but unfortunately our application for the Newsela data has not been approved.

[4]We use the original script in `https://github.com/cocoxu/simplification/blob/master/SARI.py`. All SARI values in our paper are calculated by this script.

[5]We use the script in `https://github.com/moses-smt/mosesdecoder/blob/master/scripts/generic/multi-bleu.perl`. All BLEU values in our paper are calculated by this script.

[6]We use the script in `https://github.com/XingxingZhang/dress/blob/master/dress/scripts/readability/getFKGL.py`. All FKGL values in our paper are calculated by this script.

4.4 Baselines

The main purpose of our work is to investigate the effect of context on sentence simplification, rather than propose a model that outperforms all existing models. Thus, we use four representative models as baselines, which are all trained on the training set we construct. They are also tested on the test set we construct. The four baselines are:

(1) A BiLSTM-based encoder-decoder model which is used in DRESS (Zhang and Lapata, 2017). [7]

(2) The original transformer model (Vaswani et al., 2017).

(3) The transformer model with the pointer-generator network (TP).

(4) The transformer model with the pointer-generator network and coverage mechanism (TPC).

The sentence pairs used by the baselines in the training set and test set are the same as those used by our SUC model, but do not include context.

5 Results

5.1 Automatic Evaluation

	SARI ↑	Each part of SARI			FKGL ↓	BLEU ↑
		F_{keep}	P_{del}	F_{add}		
BiLSTM-based encoder-decoder	34.06	74.19	25.94	2.06	**10.92**	59.01
Transformer	32.47	67.40	28.00	2.02	12.05	46.67
TP	32.82	77.83	18.26	2.35	13.25	52.35
TPC	27.97	**78.09**	5.71	0.01	12.01	**66.94**
SUC(ours)	**36.52**	77.54	**29.29**	**2.74**	11.37	64.41

Table 2: Results of the automatic evaluation on our test set. We report the values of SARI, each part of SARI and FKGL(lower is better) and BLEU and use **Bold** to mark the best results. We regard SARI as the most important criterion for automatic evaluation.

The results of the automatic evaluation are shown in Table 2. Our SUC model achieves 36.52 on SARI value, which outperforms the baselines. Compared with the original transformer model, our model improves on the SARI value by 4.05 points. In terms of the individual SARI values, our model also achieves the highest scores for the deleting and adding operations. For the keeping operation, our model's score is slightly lower than that of TPC that performs fewer deleting and adding operations and thus gets higher scores on keeping operations.

As for the FKGL values, although our model does not achieve the best result, our model's FKGL value is just 0.45 points higher than that of the BiLSTM-based encoder-decoder and much lower than the rest of the baselines. The TP gets the highest FKGL value but it improves the SARI value by better keeping operation compared with the transformer. As for the BLEU values, our SUC model and the best-performing TPC model yield similar results and far outperform the other three models. However, it is worth noting that the SARI value of our model is 8.55 higher than that of the TPC model.

We regard SARI as the most important criterion for automatic evaluation. Therefore, the automatic evaluation results illustrate that our model outperforms the baselines and that the context contributes to sentence simplification. Examples of the sentences generated from our model and the baseline models are given in Table 3.

5.2 Human Evaluation

In addition to automatic evaluation, we conducted the human evaluation on the outputs of different models. We randomly selected 50 sentences from the test set for evaluation.[8] Following previous works

[7] The code is available at `https://github.com/mounicam/wiki-auto/tree/master/simplification`

[8] One of the goals of text simplification is to provide convenience for non-native speakers. Therefore, we invited two non-native speakers with English proficiency roughly equivalent to 10-12 years old in English-speaking countries as volunteers. Volunteers had fully understood the evaluation criteria before conducting the evaluation and they were given complex sentences and different system outputs in random order. After the evaluation, they received a sum of money as payment.

Examples	
Original sentence	*On census night 2006 , Goulburn had a population of 20,127 people .*
Reference sentence	*In 2006 there were 20,127 people living in Goulburn .*
BiLSTM-based encoder-decoder	*on census night 2006 , goulburn had a population of 20,127 people .*
Transformer	*in 2006 there were 89 people were living in 2006 , but in fact she had a population of about 6,500 people .*
TP	*in 2006 there were 20,127 people living in the population of 20,127 people were living in 2006 .*
TPC	*on census night 2006 , goulburn had a population of 20,127 people .*
SUC(ours)	*in 2006 there were 20,127 people living in goulburn .*

Table 3: Examples of the sentences generated from our model and the baseline models. The first model and TPC do not make any changes to the original sentence. The transformer model fails to generate uncommon numbers. The TP appears to generate words repeatedly. Only our model generates the same sentence as the reference.

(Dong et al., 2019; Zhao et al., 2020), the volunteers rated the simplified sentences from the following three aspects: (1) Fluency: Is the sentence smooth and grammatical? (2) Adequacy: Is the main meaning of the original sentence retained? (3) Simplicity: Is the output simpler than the original sentence?

	Fluency	Adequacy	Simplicity	Avg
BiLSTM-based encoder-decoder	3.12	2.81	3.45	3.13
Transformer	2.67	2.70	3.38	2.92
TP	2.95	3.18	3.18	3.10
TPC	2.97	2.98	3.00	2.98
SUC(ours)	**3.36**	**3.19**	**3.54**	**3.36**
Reference	4.41	3.32	4.02	3.92

Table 4: Results of human evaluation on 50 sentence pairs randomly selected from our test set. We use **Bold** to mark the best results other than the reference. Avg represents the average scores of the three aspects.

Apart from the outputs of different models, volunteers also rated the reference. The five-point Likert scale is used for rating, and the results of the human evaluation are shown in Table 4. It can be seen that our SUC model outperforms baselines in all the aspects and the average score. Although the fluency of generated sentences of our model is still far from the reference, it exceeds that of the second-ranked model by 0.24 points. In terms of adequacy, the performance of our model is close to the reference. In terms of simplicity, our model outperforms the second-ranked model by 0.09 points, which is also the best performing model.

5.3 Ablation Study

We designed ablation experiments to explore the effects of different modules in our model on the results, especially the effects of the preceding sentences and the following sentences. We designed five additional experiments:

(1) We added an additional encoder to TPC and only used 114K sentence pairs with the preceding sentences to train the model (TPC-P).

(2) We added an additional encoder to TPC and used 114K sentence pairs with the preceding sentences and 37K sentence pairs without context to train the model (TPC-PF).

(3) We added an additional encoder to TPC. There are a total of 134K sentence pairs with the following sentences in our dataset. To better compare the impact of the position of the added contextual information, we randomly selected 114K sentence pairs with the following sentences to train the model

(TPC-F).

(4) We added an additional encoder to TPC and used 134K sentence pairs with the following sentences and 17K sentence pairs without context to train the model (TPC-FF).

(5) We added an additional encoder to TPC. We simply spliced the preceding sentences and the following sentences together and fed them into the additional encoder, which means the preceding sentences and the following sentences won't have any extra weight (TPC-S).

	SARI↑	F1-scores of SARI			FKGL↓	BLEU↑	Size of the training set
		F_{keep}	F_{del}	F_{add}			
TPC	27.97	**78.09**	5.71	0.01	12.01	**66.94**	151K
TPC-P	32.70	76.91	19.50	1.70	12.36	50.50	114K
TPC-PF	**33.01**	76.44	20.53	2.06	11.73	56.43	151K
TPC-F	32.17	75.95	18.74	1.82	11.05	66.14	114K
TPC-FF	32.94	73.23	**23.16**	**2.43**	**10.10**	61.15	151K
TPC-S	32.43	75.70	19.78	1.82	11.00	65.84	151K

Table 5: The results of ablation experiments on our test set. We use **Bold** to mark the best results.

The results of ablation experiments are shown in Table 5. From the results, we can see that when adding context, the SARI value increased by nearly five points compared to TPC, which means context is helpful for simplification. In particular, context helps immensely with deleting and adding operations, which we believe is the result of the additional information provided by the context. In experiment 2 and experiment 4, when the sentences without context information are added for training, there is a slight increase in the SARI value and a more significant decrease in the FKGL value, indicating that training with more sentence pairs can improve the readability of the output sentences. The results of experiment 5 show that simply splicing together the preceding and following sentences does not improve simplification very much, and is even less effective than using the full dataset with preceding or following sentences. This also demonstrates the necessity of treating the preceding and following sentences separately and assigning different weights to them.

6 Conclusion

In this paper, we propose a new model named SUC which makes full use of the context information for text simplification. From the results of the automatic evaluation and human evaluation, it can be seen that our model outperforms the baselines, which proves that context information is helpful to sentence simplification. In the ablation experiment, we show that sentence pairs with the preceding sentences or following sentences can both significantly improve simplification.

Acknowledgments

This work was supported by National Natural Science Foundation of China (61772036), Beijing Academy of Artificial Intelligence (BAAI) and Key Laboratory of Science, Technology and Standard in Press Industry (Key Laboratory of Intelligent Press Media Technology). We appreciate the anonymous reviewers for their helpful comments. Xiaojun Wan is the corresponding author.

References

Ziqiang Cao, Chuwei Luo, Wenjie Li, and Sujian Li. 2017. Joint copying and restricted generation for paraphrase. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, pages 3152–3158. AAAI Press.

William Coster and David Kauchak. 2011. Simple english wikipedia: a new text simplification task. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 665–669.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American*

Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 4171–4186.

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

John Duchi, Elad Hazan, and Yoram Singer. 2011. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research*, 12(7).

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

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

Dan Hendrycks and Kevin Gimpel. 2016. Bridging nonlinearities and stochastic regularizers with gaussian error linear units. *arXiv preprint arXiv:1606.08415*.

Sébastien Jean, Orhan Firat, Kyunghyun Cho, Roland Memisevic, and Yoshua Bengio. 2015. Montreal neural machine translation systems for wmt'15. In *Proceedings of the Tenth Workshop on Statistical Machine Translation*, pages 134–140.

Chao Jiang, Mounica Maddela, Wuwei Lan, Yang Zhong, and Wei Xu. 2020. Neural crf model for sentence alignment in text simplification. *arXiv preprint arXiv:2005.02324*.

Hanqi Jin, Tianming Wang, and Xiaojun Wan. 2020. Semsum: Semantic dependency guided neural abstractive summarization. In *AAAI*, pages 8026–8033.

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

David Kauchak. 2013. Improving text simplification language modeling using unsimplified text data. In *Proceedings of the 51st annual meeting of the association for computational linguistics (volume 1: Long papers)*, pages 1537–1546.

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. Technical report, Naval Technical Training Command Millington TN Research Branch.

Reno Kriz, João Sedoc, Marianna Apidianaki, Carolina Zheng, Gaurav Kumar, Eleni Miltsakaki, and Chris Callison-Burch. 2019. Complexity-weighted loss and diverse reranking for sentence simplification. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3137–3147.

Piji Li, Wai Lam, Lidong Bing, and Zihao Wang. 2017. Deep recurrent generative decoder for abstractive text summarization. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2091–2100.

Shuming Ma and Xu Sun. 2017. A semantic relevance based neural network for text summarization and text simplification. *arXiv preprint arXiv:1710.02318*.

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

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

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

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

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

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Ellie Pavlick and Chris Callison-Burch. 2016. Simple ppdb: A paraphrase database for simplification. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 143–148.

Emily Pitler and Ani Nenkova. 2008. Revisiting readability: A unified framework for predicting text quality. In *Proceedings of the 2008 conference on empirical methods in natural language processing*, pages 186–195.

Horacio Saggion. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Abigail See, Peter J Liu, and Christopher D Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083.

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

Elior Sulem, Omri Abend, and Ari Rappoport. 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. Semantic structural evaluation for text simplification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 685–696.

Zhaopeng Tu, Zhengdong Lu, Yang Liu, Xiaohua Liu, and Hang Li. 2016. Modeling coverage for neural machine translation. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 76–85.

Zhaopeng Tu, Yang Liu, Shuming Shi, and Tong Zhang. 2018. Learning to remember translation history with a continuous cache. *Transactions of the Association for Computational Linguistics*, 6:407–420.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008.

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

Lesly Miculicich Werlen, Dhananjay Ram, Nikolaos Pappas, and James Henderson. 2018. Document-level neural machine translation with hierarchical attention networks. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2947–2954.

Kristian Woodsend and Mirella Lapata. 2011. Wikisimple: automatic simplification of wikipedia articles. In *Proceedings of the Twenty-Fifth AAAI Conference on Artificial Intelligence*, pages 927–932.

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

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.

Jiacheng Zhang, Huanbo Luan, Maosong Sun, Feifei Zhai, Jingfang Xu, Min Zhang, and Yang Liu. 2018. Improving the transformer translation model with document-level context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 533–542.

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.

Yanbin Zhao, Lu Chen, Zhi Chen, and Kai Yu. 2020. Semi-supervised text simplification with back-translation and asymmetric denoising autoencoders. In *AAAI*, pages 9668–9675.

Yang Zhong, Chao Jiang, Wei Xu, and Junyi Jessy Li. 2020. Discourse level factors for sentence deletion in text simplification. In *Proceedings of the Thirty-Fourth AAAI Conference on Artificial Intelligence*.

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