Natural Language Engineering (2024), **30**, pp. 915–942
doi:10.1017/S1351324923000566

ARTICLE

How do control tokens affect natural language generation tasks like text simplification

Zihao Li and Matthew Shardlow

Manchester Metropolitan University, Manchester, UK
Corresponding author: Zihao Li; Email: 21443696@stu.mmu.ac.uk

(Received 27 July 2023; revised 12 December 2023; accepted 12 December 2023; first published online 23 January 2024)

Abstract
Recent work on text simplification has focused on the use of control tokens to further the state-of-the-art. However, it is not easy to further improve without an in-depth comprehension of the mechanisms underlying control tokens. One unexplored factor is the tokenization strategy, which we also explore. In this paper, we (1) reimplemented AudienCe-CEntric Sentence Simplification, (2) explored the effects and interactions of varying control tokens, (3) tested the influences of different tokenization strategies, (4) demonstrated how separate control tokens affect performance and (5) proposed new methods to predict the value of control tokens. We show variations of performance in the four control tokens separately. We also uncover how the design of control tokens could influence performance and give some suggestions for designing control tokens. We show the newly proposed method with higher performance in both SARI (a common scoring metric in text simplificaiton) and BERTScore (a score derived from the BERT language model) and potential in real applications.

Keywords: Text simplification; controlable text simplification

1. Introduction

Text simplification (TS) mainly refers to the process of reducing linguistic complexity at both syntactic and lexical levels and improving readability and understandability (Devlin 1998; Shardlow 2014; Alva-Manchego, Scarton, and Specia 2020b). It is commonly used to increase the readability of documents intended for children (De Belder and Moens 2010), non-native speakers (Petersen and Ostendorf 2007) and people with dyslexia. The requirements for simplified outcomes may vary among audiences (Xu, Callison-Burch, and Napoles 2015), for instance, depending on the characteristics of the dataset. The task of simplification has been approached through sentence-level simplification (Nishihara, Kajiwara, and Arase 2019; Martin *et al.* 2020a) and paragraph-level simplification (Sun, Lin, and Wan 2020; Devaraj *et al.* 2021). In this paper, we focus only on sentence-level simplification, as this is where the majority of work on control tokens exists.

In order to fit the requirements of different user groups, some projects introduced explicit discrete prompts as control tokens to assist TS models to learn specific properties from datasets and adjust the resultant simplifications accordingly (Martin *et al.* 2020a; Agrawal, Xu, and Carpuat, 2021). By adjusting the value in different control tokens, researchers can manually adjust the characteristics of the output, such as length, syntactic and lexical difficulty, etc.

The control tokens are added to the beginning of the complex sentences and represent a relationship between that sentence and the desired input (such as the desired compression ratio). In addition, the numerical value also changes with the demands of the outcome. The format of the

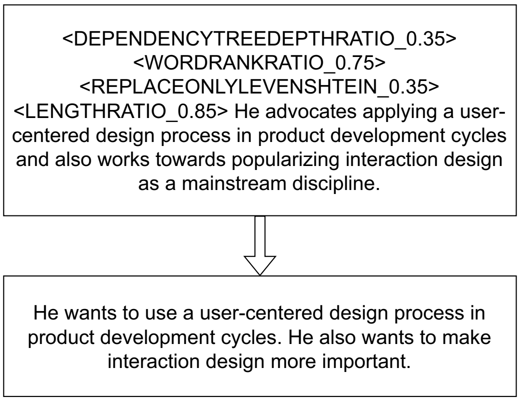
<DEPENDENCYTREEDEPTHRATIO_0.35> <WORDRANKRATIO_0.75> <REPLACEONLYLEVENSHTEIN_0.35> <LENGTHRATIO_0.85> He advocates applying a user-centered design process in product development cycles and also works towards popularizing interaction design as a mainstream discipline.
He wants to use a user-centered design process in product development cycles. He also wants to make interaction design more important.

Figure 1. An instance of preprocessed input and raw output of the system.

control token is <**Token**_*value*>, where **Token** is a novel extra-vocabulary token with human interpretable meaning, and *value* is a numerical value indicating some relationship between the given input and output as shown in Fig. 1 and discussed Section 5.1. The design of the control tokens is based on the need for adjustment. Multiple control tokens can be applied simultaneously, and four control tokens are used in this project.

Although the control tokens are manually crafted, how the control tokens change the outcome remains unstudied. To explore the mechanisms of control tokens in simplification, this paper proposes the following: (1) Reimplement ACCESS (Martin *et al.* 2020a), the state-of-the-art (SOTA) at the time of writing in Section 4.3. (2) Verify the importance of control tokens in Section 5. (3) Explore the influence and interactions of the variation of control tokens in the format in Section 5.1. (4) Investigate the effects of the tokenization method in Section 5.2. (5) Propose both regressive and multi-classification methods on the prediction of the value of control tokens and compare the performance difference with existing methods in Sections 5.3, 5.4 and 5.5.[a] (6) Study example cases to evaluate the newly proposed methods in Section 6.2.

Among the above-mentioned content, we will focus on the following three research questions:

- Whether the quantisation of applying control tokens will significantly affect the performance;
- Whether the tokenization method of control tokens will significantly affect the performance;
- Whether the control token predictor can predict the ideal values and hence significantly improve the performance.

2. Literature review

2.1. General text simplification

Natural language generation (NLG) is a sub-task in natural language processing. There have been attempts to build an NLG system based on hand-crafted rules and to define the problem and features based on knowledge in the last century (Hovy 1990; Reiter and Dale 1997). With the development of computation power and the re-introduction of neural networks, more neural-network-based statistical methods were applied (Wen *et al.* 2015; Dušek and Jurčíček 2016; Lebret, Grangier, and Auli 2016; Mei, Bansal, and Walter 2016). One important change happened with

[a]All code will be available at https://github.com/lizihao041/Investigation-of-control-token

the publishing of the transformer architecture (Vaswani *et al.* 2017), which inspired the "pre-train and fine-tune" paradigm. Later, due to the outstanding performance of this new architecture in both performance and computation consumption, the transformer architecture and its derivatives occupied a dominant position in the NLG domain (Yang *et al.* 2019; Floridi and Chiriatti 2020; Lewis *et al.* 2020). As a sub-task of NLG, TS can also be regarded as monolingual machine translation (Wubben, van den Bosch, and Krahmer 2012). With the development of sequence-to-sequence machine translation, TS also drew more attention (Guo, Pasunuru, and Bansal 2018; Surya *et al.* 2019; Omelianchuk, Raheja, and Skurzhanskyi 2021).

2.2. Controllable text simplification

In recent years, researchers introduced explicit parameters to control the simplified output (Nishihara *et al.* 2019; Martin *et al.* 2020a; Agrawal *et al.* 2021). Martin *et al.* (2020a) introduced four hyper-parameters in the AudienCe-CEntric Sentence Simplification (ACCESS): the number of characters, Levenshtein similarity (Levenshtein *et al.* 1966), word rank and dependency tree depth, which are used to control the length, similarity, lexical complexity and syntactic complexity, respectively. With the help of the parameters, users can modify the generated simplification based on their needs. However, these parameters may be less straightforward for untrained users, and Agrawal *et al.* (2021) replaced the detailed parameters with simplification grades. In addition, a minor change in these parameters may significantly affect the readability and fluency of output. Although the value set that maximises the benchmark scores can be given, it may be of little help to the end-users with specific requirements. Further exploration of the effect and proper parameter preferences needs to be made to guide and help untrained users adjust these parameters based on their needs.

Another novel research on the training datasets is multilingual unsupervised sentence simplification (MUSS) (Martin *et al.* 2020b). They fine-tuned Bidirectional and Auto-Regressive Transformer (BART) (Lewis *et al.* 2020) on their mined paraphrases datasets instead of complex-simple parallel corpora and found that with the help of ACCESS, the unsupervised model outperformed the other unsupervised TS models and became the latest SOTA. As an extension of ACCESS, the authors improved the design of control tokens and changed the tokenization strategy. They showed that performance differences between the two types of datasets might be acceptable only if the mined paraphrase dataset is good enough. Training on paraphrase datasets provides more options than training solely on the supervised datasets and there is a nearly unlimited amount of unlabelled data. They also found that the performance of the combination of unsupervised and supervised training is the best, which is very similar to the pre-train and fine-tune paradigm. Although MUSS also includes multilingual models and tests, the models of different languages were trained separately and had little interference with the aim of this project. Thus, there is no need to focus on other languages for our study of control tokens.

2.3. Metrics

The metrics also play a vital role in evaluating the performance of models. Although current metrics do not compete with human evaluations, they can still partially reflect the performance in certain indexes. Among the popular metrics, there are reference-based metrics like Bilingual evaluation understudy (Papineni *et al.* 2002) and Recall-Oriented Understudy for Gisting Evaluation (Lin 2004) and non-reference-based metrics like Flesch-Kincaid Grade Level (Flesch 1948). Currently, the most popular metric for TS is the system output against references and the input sentence (SARI) (Xu *et al.* 2016). SARI is designed especially for TS tasks, which evaluates the outputs in aspects of adding, keeping and deleting. Although it is found to have some deviation from human judgement, SARI is still a valuable metric to evaluate simplicity (Alva-Manchego, Scarton, and Specia 2021). As for the non-reference-based metrics, the BERTScore is a BERT-based metric

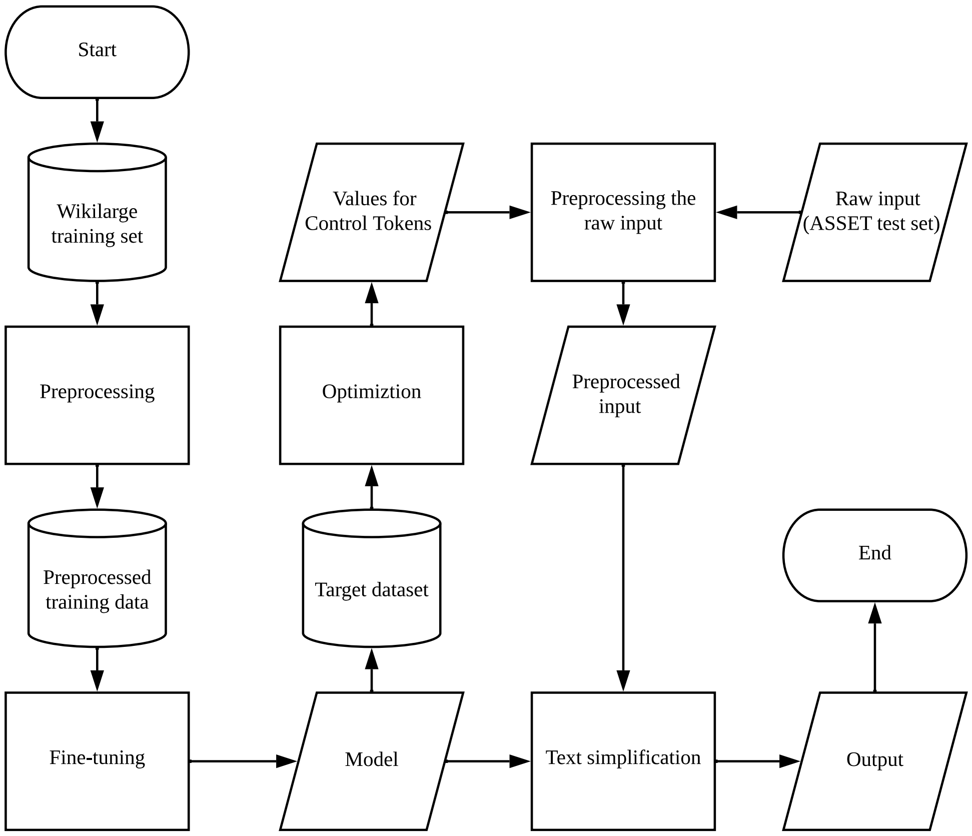

Figure 2. The reimplementation methodology is represented as a flow chart. We fine-tune BART-base on the preprocessed WikiLarge training set so that the model learns to simplify under the control token or control tokens. After optimisation on the target dataset, the model can apply the optimal value of control tokens to the input and generate desired simplifications.

that evaluates the similarity between the output and references by calculating the correlation in the embedding space (Zhang *et al.* 2019). It is found to have a high correlation with human judgement (Scialom *et al.* 2021). By combining the metrics, the performance can be evaluated more comprehensively.

3. Methodology

In this section, we demonstrate the methodology in Fig. 2. There are 4 main steps in the whole system: preprocessing, fine-tuning, optimisation and evaluation.

3.1. Preprocessing

The preprocessing step followed the MUSS project (Martin *et al.* 2020b). The authors defined four types of prompts used as control tokens to manipulate the features of the outputs. Each control token is designed to represent one character of the sentence. The <DEPENDENCY TREEDEPTH_x> represents the syntactic complexity; The <WORDRANK_x> represents the lexical complexity; The <REPLACEONLYLEVENSHTEIN_x> represents the inverse similarity of input and output at the letter level; The <LENGTHRATIO_x> represents the length ratio of input and output. The value of each control token is calculated based on the reference complex-simple pairs in the training dataset, which is WikiLarge in this project (Zhang and Lapata 2017). The WikiLarge dataset (Zhang and Lapata 2017) is one of the biggest parallel complex-simple sentence datasets based on various existing corpora and contains 296,402 sentence pairs in the training set. After the calculation, these control tokens will be added to the beginning of complex sentences, and the model will be trained on this preprocessed dataset. In addition to the combined control tokens, this project also explored the effects of a single-control token. In these experiments, only the corresponding control tokens are kept in that dataset.

3.2. Fine-truning

The next step is training. It follows the majority of fine-tuning processes for pre-trained language models. By feeding the preprocessed complex-simple sentence pairs to the model, the model is expected to learn how to simplify texts based on the instruction of control tokens. In our experiments, we explore the use of different tokenization strategies for the control tokens, proposing three possible methods. We also explore the contribution of individual control tokens under each tokenization strategy. In total, 16 models are fine-tuned in the experiment: 1 baseline model, 1 model with all control tokens for each tokenization strategy (3 models) and a separate model for each of our four control tokens under each tokenization strategy (12 models).

3.3. Optimisation

The next step is optimisation. As mentioned in previous sections, the value of control tokens is limited to a small range. All options fall between 0.2 to 1.5 except the Levenshtein, whose upper boundary is limited to 1 due to the calculation method that divides the minimum replacement steps to change from the original sentence to the target sentence by the maximum possible steps of replacement. Only these options are provided during optimisation, and the optimisation problem is reduced to finding the best value combination of control tokens within the optimisation budget. Even though only finite combinations can be applied to the model, the optimisation algorithm is still supported by the Nevergrad (Rapin and Teytaud 2018) API to compare with the current SOTA. Within budget as a limitation to repeat the optimisation process 64 times, the algorithm can find a relatively optimised result in all 334,611 combinations of control token values. Although we have half the optimisation budget compared to MUSS, the system still manages to achieve better performance than MUSS. A sample result of SARI in the optimisation procedure is shown in Appendix A.1. In order to ensure the reliability of the score under the optimised combination, a bootstrapping on the Abstractive Sentence Simplification Evaluation and Tuning (ASSET) (Alva-Manchego *et al.* 2020a) test dataset will be executed by resampling the dataset 200 times and hence generate a 95% confidence interval.

3.4. Evaluation

The last step is evaluation. Thanks to Easier Automatic Sentence Simplification Evaluation, multiple evaluation metrics can be applied at the same time easily (Alva-Manchego *et al.* 2019). The SARI score is adopted as the primary metric to compare with the current SOTA, while the BERTScore is added as a second reference. Different from the common applications in other projects, the BERTScore in this project is the correlation between the output and references. One coefficient array can be used to combine different evaluation metrics and give a weighted score. However, in this project, we also follow the operations in MUSS and maximise the SARI score, so only the SARI score is taken into account, and the corresponding coefficient is set to 1. The models will be evaluated on the ASSET (Alva-Manchego *et al.* 2020a) test dataset, which contains 359 complex-simple pairs, and each complex sentence has ten reference simplifications.

3.5. Metrics

SARI is an operation-specific metric for evaluating the performance of the simplicity gained through TS system (Xu *et al.* 2016). It offers a comprehensive evaluation by comparing the system output to both the reference and input sentences and considering the *add*, *keep*, and *delete* operations. The final SARI score is calculated in Equation (1) as follows:

$$SARI = d_1 F_{\text{add}} + d_2 F_{\text{keep}} + d_2 F_{\text{del}} \tag{1}$$

where

$$d_1 = d_2 = d_3 = 1/3 \tag{2}$$

and

$$P_{\text{operation}} = \frac{1}{k}\sum_{n=1}^{k} \mathrm{p}_{\text{operation}}(n) \quad R_{\text{operation}} = \frac{1}{k}\sum_{n=1}^{k} \mathrm{r}_{\text{operation}}(n)$$

$$F_{\text{operation}} = \frac{2 \times P_{\text{operation}} \times R_{\text{operation}}}{P_{\text{operation}} + R_{\text{operation}}} \quad \text{operation} \in \{\text{del, keep, add}\} \tag{3}$$

In Equation (1), the different operations have different algorithms, which take into account output (O), the input sentence (I), references (R), and the binary indicator for the occurrence of n-grams g in a given set ($\#_g(.)$). The n-gram precision $p(n)$ and recall $r(n)$ are calculated in the following equations:

$$\begin{aligned}
p_{\text{add}}(n) &= \frac{\sum_{g\in O} \min\left(\#g(O\cap \bar{I}), \#g(R)\right)}{\sum_{g\in O} \#g(O\cap \bar{I})}, \quad \#g(O\cap \bar{I}) = \max\left(\#g(O) - \#g(I), 0\right); \\
r_{\text{add}}(n) &= \frac{\sum_{g\in O} \min\left(\#g(O\cap \bar{I}), \#g(R)\right)}{\sum_{g\in O} \#g(R\cap \bar{I})}, \quad \#g(R\cap \bar{I}) = \max\left(\#g(R) - \#g(I), 0\right), \\
p_{\text{keep}}(n) &= \frac{\sum_{g\in I} \min\left(\#g(I\cap O), \#g(I\cap R_0)\right)}{\sum_{g\in I} \#g(I\cap O)}, \quad \#g(I\cap O) = \min\left(\#g(I), \#g(O)\right), \\
r_{\text{keep}}(n) &= \frac{\sum_{g\in I} \min\left(\#g(I\cap O), \#g(I\cap R_0)\right)}{\sum_{g\in I} \#g(I\cap R_0)}, \quad \#g(I\cap R_0) = \min\left(\#g(I), \#g(R)/r\right), \\
p_{\text{del}}(n) &= \frac{\sum_{g\in I} \min\left(\#g(I\cap \overline{O}), \#g(I\cap \overline{R_0})\right)}{\sum_{g\in I} \#g(I\cap \overline{O})}, \quad \#g(I\cap \overline{O}) = \max\left(\#g(I) - \#g(O), 0\right), \\
&\qquad \#g(I\cap \overline{R_0}) = \max\left(\#g(I) - \#g(R)/r, 0\right).
\end{aligned} \tag{4}$$

The other automatic metric we applied is the BERTScore (Zhang *et al.* 2019), which is a metric used for evaluating the similarity between the system output and reference sentences. This metric employs cosine similarity to measure the distance in contextual embeddings created by Bidirectional Encoder Representations from Transformers (BERT) (Devlin *et al.* 2019) in a likelihood matrix, applying greedy matching to maximise the matching similarity score. BERTScore comprises two components: $BERTScore_{precision}$, which matches the system output with the reference, and $BERTScore_{recall}$, which compares the reference against the system output. The two parts

Table 1. Tokenization under differing strategies for the input starting with: '<DEPENDENCYTREEDEPTHRATIO_0.6>'

Strategy	Raw input	'< DEPENDENCYTREEDEPTHRATIO_0.6>'
Default	IDs	[0, 41552, 41372, 9309, 23451, . . ., 2571, 6454, 1215, 288, 4, . . .]
	tokenization	['< s >', '<', 'DEP', 'END', 'ENCY', . . ., '_', '0', '.', '6', '>', . . .]
Joint	IDs	[0, 50265, . . .]
	tokenization	['< s >', '< DEPENDENCYTREEDEPTHRATIO_0.6 >', . . .]
Separate	IDs	[0, 50265, 50266, 15698, . . .]
	tokenization	['< s >', '< DEPENDENCYTREEDEPTHRATIO_', '0.6', '>', . . .]

are then combined in $BERTScore_{F1}$ as shown in Equation (5), where x and $\hat{x}$ are the references and candidates.

$$BERTScore_{recall} = \frac{1}{|x|} \sum_{x_i \in x} \max_{\hat{x}_j \in \hat{x}} x_i^\top \hat{x}_j,$$

$$BERTScore_{precision} = \frac{1}{|\hat{x}|} \sum_{\hat{x}_j \in \hat{x}} \max_{x_i \in x} x_i^\top \hat{x}_j,$$

$$BERTScore_{F1} = \frac{2 \cdot BERTScore_{precision} \cdot BERTScore_{recall}}{BERTScore_{precision} + BERTScore_{recall}} \tag{5}$$

4. Experiment design

In this section, we will focus on the detailed settings and changes in this project compared to the original ACCESS (Martin *et al.* 2020a) and the experiment settings for the above-mentioned research questions.

4.1. Tokenization strategies

We introduced two new tokenization methods (compared to the original implementation of ACCESS/MUSS) to explore the effects of tokenization strategies. As shown in Table 1, the default tokenization method in the MUSS project is regarding the control tokens as plain text. In comparison, we added 2 more tokenization strategies: The Joint tokenization strategy is to regard the whole control token as one token in the tokenizer; the Separate tokenization strategy is to break the control token into a combination of type and value and add them separately to the tokenizer. These 2 strategies are achieved by manually adding all possible control tokens to the dictionary of the tokenizer. This will affect not only the evaluation and optimisation process but also the training process, thus each tokenization strategy requires an independent fine-tuned model.

4.2. Quantisation strategy

We implemented a quantisation strategy for the control tokens in the optimisation step and the corresponding algorithm. In the preprocessing step, there are 4 control tokens calculated and added to the beginning of complex sentences in the complex dataset. As mentioned in the literature review, the 4 types of control tokens are <DEPENDENCYTREEDEPTH_x> (DTD), <WORDRANK_x> (WR), <REPLACEONLYLEVENSHTEIN_x> (LV) and <LENGTHRATIO_x> (LR). As an augmentation to the control tokens, the calculated values are rounded to the nearest 0.05. However, in the optimisation process as originally proposed by Martin *et al.* (2020a), the calculated values by the algorithm provided by the Nevergrad API (Rapin and Teytaud 2018) are continuous and have long verbose digits, 0.249452. . .for example. During our reimplementation, we noticed that only the first one or two digits were recognised as input values and the remaining digits did not provide any meaningful instruction. On the contrary, it brought unnecessary information to the system and even lowered the performance of the model. Thus we replaced the continuous values with discrete values (0.2, 0.25, 0.3, . . ., 1.0) and changed to the corresponding discrete algorithm in Nevergrad (Rapin and Teytaud 2018).

4.3. Prediction of optimal control tokens

In contrast to the method of finding an optimised set of control tokens at the corpus level adopted in MUSS (Martin *et al.* 2020b), we propose both regressive and multi-classification methods of predicting the value of each control token at the sentence level. This method is based on two hypotheses: There is a unique ideal simplification for every sentence; By feeding the ideal value of control tokens from the ideal simplification, the model can generate the ideal simplification. Unlike optimising the value of control tokens on the test set, we assume the ideal value is an intrinsic property and can be predicted. In addition, the problem can be regarded as a regression problem to predict the optimal value of control tokens and how to fine-tune autoregression models to do so for every sentence. In the experiment, we chose the average value of multiple reference sentences as an approximation to the ideal value and compared it with the predicted values and the optimised value. Although the median value of reference sentences shows slightly higher SARI and BERTScore on the ASSET test set, the predicted median values are not as good as the calculated values. In addition, both the average and median meet the requirements as an approximation of ideal values and choosing one or the other makes no difference in the conclusion.

In the training process of predictors for each control token, we fine-tuned the BERT-base-uncased model on the filtered WikiLarge dataset (Zhang and Lapata 2017), targeting the common end-users and the ASSET test set as well. We filter the sentences with values in the range of 0.2–1.5 and keep the model with the lowest root mean square error within 10 epochs. For each control token, we report the normalised mean absolute error (MAE) and root mean square error (RMSE). We show the distribution difference with the average value or the expectations and values from all reference sentences on the ASSET test set. In comparison to the optimisation method, we test the predictor model with the corresponding single-control token models, which are fine-tuned in previous sections, on SARI and BERTScore. Then we assembled and tested them as a whole system and reported the SARI and BERTScore.

4.4. Reimplementation of ACCESS

We focus our reimplementation on the ACCESS model (Martin *et al.* 2020a). Although MUSS (Martin *et al.* 2020b), outperforms ACCESS, it does so by using additional training data, as opposed to any innovation concerning the control tokens. As such, we do not use the additional mined paraphrases as used in MUSS for training.

Whereas ACCESS uses the original base Transformer architecture, we use BART as our base model (Lewis *et al.* 2020), which is also adopted in the MUSS project. The original project can be divided into the following sections: data mining, preprocessing, training, evaluation and optimisation.

The original library used for training is fairseq. This project replaced it with another open-source library—Huggingface. Huggingface provides a collection of the most popular pre-trained models and datasets, including BART (Lewis *et al.* 2020) and a unified, advanced and user-friendly API to achieve the most common applications, which makes it easier for future upgrading and modification. The hyper-parameters of models in the reimplementation, including the learning rate and weight decay, are set to be identical to the original project so that the influence of irrelevant factors can be lowered. The last difference between the reimplementation and the original project is the tokeniser. The tokeniser in the reimplementation is the BART-base byte-pair encoding (BPE) tokeniser instead of the GPT2 BPE tokeniser (Radford *et al.* 2019). Both tokenisers serve the same purpose and perform very similarly to each other. The new one consumes fewer computing resources, which presumably causes only a little effect on the results. Due to the variation of control tokens, the optimisation algorithm has also changed. The original algorithm is the OneplusOne provided by Nevergrad (Rapin and Teytaud 2018), and the current one is the PortfolioDiscreteOnePlusOne, which fits the discrete values better. As for metrics, the SARI score is kept as the primary evaluation method (Xu *et al.* 2016), and BERTScore is introduced as a co-reference.

However, due to the limitation of computation resources and mass fine-tuning demands of models with different tokenization strategies, this project also downgraded the training scale and limited the epochs in both baseline and reimplementation. Here are the changes applied to both the reimplementation and the baseline as follows:

- All results are from models trained in BART-base instead of BART-large;
- All training processes are set to 10 epochs only;
- All models are trained on WikiLarge (Zhang and Lapata 2017) only.

As explained earlier, each tokenization strategy corresponds to one model and 16 models in total need to be fine-tuned. This is why only BART-base is applied and the training epochs are limited. The reason for choosing 10 as the targeting epoch number is because the training loss for models with combined control tokens has reached 0.85 and decreased very slowly between epochs, while the validation loss started increasing. If continuing training, the over-fitting problem may occur.

5. Results

We firstly report the SARI score of influential supervised TS models with our reimplementations in Table 2. Although MUSS (with mined data) (Martin *et al.* 2020b) is slightly lower than our reimplementation, our reimplementation stays within the 95% confidence interval of MUSS (with mined data). To verify the significance of the difference in the SARI score, we conducted significance studies against the official output of MUSS (without mined data) with a student's t-test of the SARI score of the two groups and reported the p-value for the bottom four models. As shown in the table, our reimplementation required fewer resources and training data, while maintaining a significant difference. In addition, we propose the prediction rather than optimisation and get a higher BERTScore in Section 5.4. The bottom two methods in Table 2 are described in Section 5.5.

5.1. Combined performance

The baseline we obtained by rerunning the original code of the current SOTA is 43.83 on the ASSET (Alva-Manchego *et al.* 2020a) test dataset. This value is consistent with our reported score on MUSS without minded paraphrasing data, which is 43.63±0.71. The reason we compare this model is that this is the test scenario closest to our reimplementation. There is no confidence interval and BERTScore in the baseline because the baseline is generated by rerunning the code in MUSS by altering specific settings only. The actual output lacks these 2 features. As shown in the 4 rows in Table 3, the SARI score with 95% confidence in the reimplementation is slightly higher than the baseline. However, significance testing shows that the improvement in score according to the different tokenization strategies is not significantly different from the baseline result for any of the cases we examined. This implies that tokenization strategy does not affect control token performance.

Table 2. The SARI score of supervised text simplification systems (p in the brackets refers to the p-value of the SARI score against the multilingual unsupervised sentence simplification (without mined data))

Model	SARI score ↑
EditNTS (Dong *et al.* 2019)	34.95
Dress-LS (Zhang and Lapata 2017)	36.59
DMASS-DCSS (Zhao *et al.* 2018)	38.67
ACCESS (Martin *et al.* 2020a)	40.13
TST (Omelianchuk *et al.* 2021)	43.21
MUSS (without mined data) (Martin *et al.* 2020b)	43.63
MUSS (with mined data) (Martin *et al.* 2020b)	44.15 ($p = 0.059$)
Our Reimplementation	44.65 ($p = 0.033$)
Our Reimplementation (predictors only)	42.30 ($p = 0.133$)
Hybrid Method	44.61 ($p = 0.056$)

Table 3. Results on SARI and BERTScore under differing tokenization strategies, with comparison to the baseline (*p* in the brackets refers to the *p*-value of the SARI score against the baseline and shows no statistical significance improvement)

Prompts	SARI↑	BERTScore↑	DTD	WR	LV	LR
Baseline (Rerun)	43.83	—	0.249452...	0.814345...	0.758764...	0.858546...
Default	44.00±0.05 ($p = 0.620$)	0.754	0.25	0.8	0.75	0.85
Joint tokens	44.02±0.05 ($p = 0.700$)	0.769	0.25	0.8	0.75	0.85
Separate tokens	44.04±0.05 ($p = 0.779$)	0.754	0.25	0.8	0.75	0.85

Different from the optimisation target in MUSS, we tried optimising control tokens on both the validation and test set of ASSET to find out the peak performance of the system and the results are shown in Table 4. The top 3 rows show the best SARI score with optimised options of control tokens optimised on the validation set. With optimised control tokens on the validation set, the separate tokenization strategy achieved the highest score within the optimisation budgets, while the joint tokenization method has the highest BERTScore. The middle 3 rows are the results directly optimised on the test set, which shows the upper limit of the model. Among the 3 methods, the separate tokenization strategy had the highest SARI score. Interestingly, the BERTScore is not always proportional to the SARI score, but the BERTScore of optimal value is still quite high. The optimised values of control tokens are pretty close in all situations except the DTD. Similar to the results in Table 3, the p-values show no significant difference between the baseline and the reimplementations. We continue to record the results for each of our three tokenization strategies, however, significance testing again shows that changes in the tokenization strategy have not led to gains which are significantly improved from the baseline in these cases. The bottom 3 rows show the performance difference under a unified value of control tokens. The unified value is the average value of all possible values for each control token. Under the unified condition, the separated one outperformed the other two, and the default tokenization method still performs worse. As for the BERTScore, the joint tokenization method still outperforms the other two.

Table 4. SARI and BERTScore on the ASSET test set under different optimisation targets (p in the brackets refers to the p-value of the SARI score against the baseline and shows no statistical significance improvement). The first three rows optimise on the validation set, the middle three rows optimise on the test set and the bottom three rows show the performance under average value of control tokens

Optimisation target	Prompts	SARI↑	BERTScore↑	DTD	WR	LV	LR
	Default	44.31±0.05 ($p = 0.628$)	0.759	0.4	0.6	0.8	1.1
Validation set	Joint tokens	44.58±0.05 ($p = 0.885$)	0.814	0.35	0.7	0.85	0.9
	Separate tokens	44.65±0.05 ($p = 0.701$)	0.783	0.35	0.75	0.75	0.95
	Default	44.36±0.05 ($p = 0.931$)	0.733	0.6	0.7	0.65	0.85
Test set	Joint tokens	44.67±0.05 ($p = 0.313$)	0.786	0.35	0.75	0.75	0.9
	Separate tokens	44.75±0.05 ($p = 0.321$)	0.784	0.35	0.75	0.75	0.9
	Default	43.34±0.06	0.827	0.6	0.85	0.85	0.85
N/A	Joint tokens	43.83±0.06	0.829	0.6	0.85	0.85	0.85
	Separate tokens	43.99±0.06	0.828	0.6	0.85	0.85	0.85

Table 5. Results on SARI and BERTScores of peak points in different control tokens (We choose the points with the highest SARI score in the three strategies and corresponding points in the other two strategies)

DTD	Strategy	SARI	BERTScore	WR	Strategy	SARI	BERTScore
	Default	40.82 ±0.05	0.805		Default	40.61±0.06	0.720
0.55	Separate	40.68±0.06	0.804	0.75	Separate	41.08±0.06	0.738
	Joint	40.71±0.06	0.812		Joint	**41.42±0.06**	0.733
	Default	40.54 ±0.05	0.799		Default	40.80±0.06	0.776
0.6	Separate	**40.87±0.05**	0.801	0.8	Separate	40.32±0.05	0.797
	Joint	40.43±0.06	0.800		Joint	40.43±0.06	0.782

LV	Strategy	SARI	BERTScore	LR	Strategy	SARI	BERTScore
	Default	42.52±0.06	0.750		Default	40.15±0.06	0.758
0.65	Separate	42.55±0.06	0.747	0.6	Separate	40.25±0.06	0.760
	Joint	42.63±0.06	0.761		Joint	40.46±0.05	0.758
	Default	42.26±0.08	0.785		Default	39.91±0.05	0.782
0.7	Separate	**42.86±0.06**	0.782	0.65	Separate	40.27±0.05	0.781
	Joint	42.31±0.07	0.787		Joint	**40.64±0.05**	0.785

5.2. Effects of single-control tokens

In order to verify the effects of each single-control token, a more detailed investigation of the SARI score was done on control tokens, respectively, and the results are shown in Table 5 and Fig. 3. Except for Fig. 3b, all 3 tokenization methods show a high consistency in the curves and

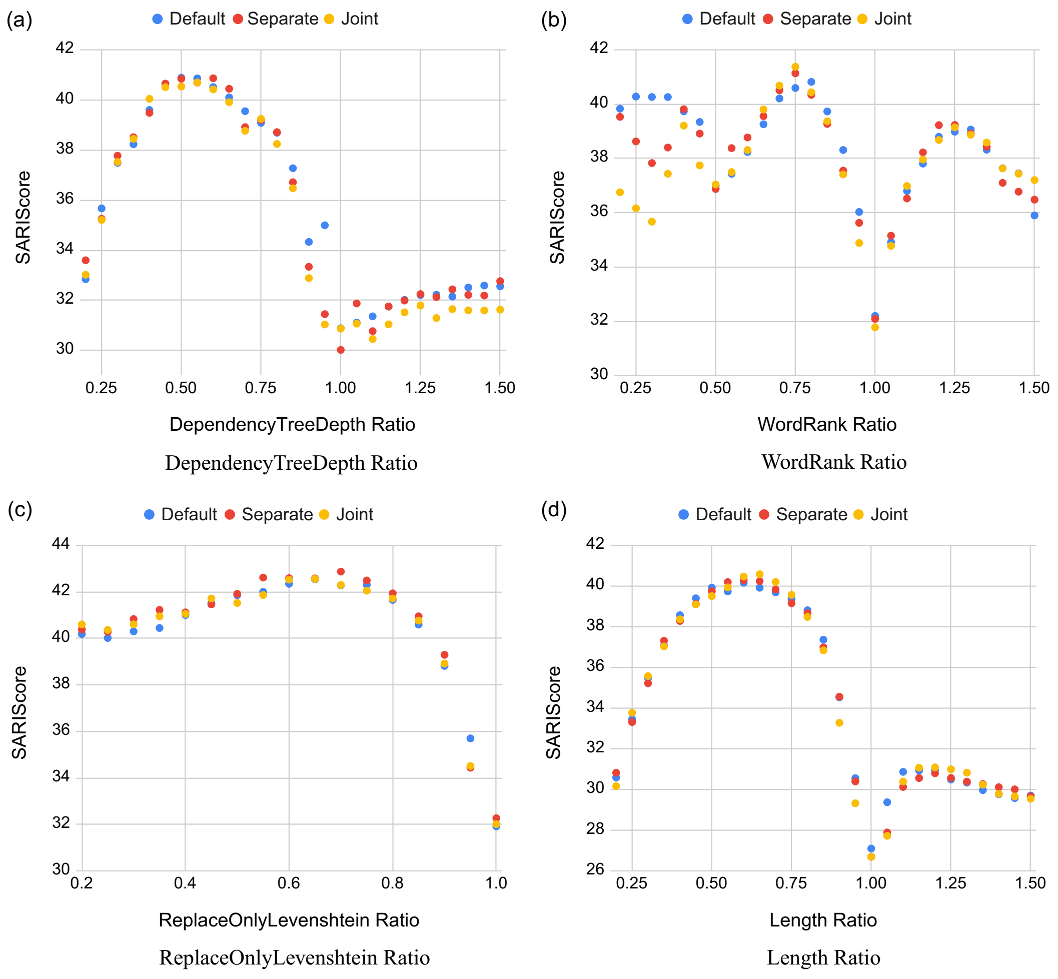

Figure 3. The effect of varying control tokens with different tokenization strategies on SARI Score.

have a common minimum at the value of 1. As shown in Table 6, it is mainly caused by the low score in both deletion and adding operations.

Table 5 shows the scores in pairs under a unified value. Similar to the combined control tokens, although the optimised SARI score of either separate or joint tokenization methods is slightly higher than the default tokenization method, there is no significant improvement in the peak performance.

In addition to the peak points, the differences in tokenization methods show little effect on the scores while the value of control tokens can change the performance significantly in the whole curves. In Fig. 3a and c, the separate tokenization method shows the highest peak point, while in Fig. 3b and d, the joint tokenization method has the best performance. More about the BERTScore is shown in Fig. A1 in Appendix A.2.

The Table 6 is designed to help readers better understand the reason for variations in Fig. 3. It shows some local minimum or maximum points within the domain and the corresponding SARI score by operations. The addition score is much lower than the keeping and deletion. It is because there is only limited adding operation in the references and many more expression options to carry a similar meaning, which leads to a low hit rate of the addition operation. At the same time, the keep and deletion are chosen from the existing input and thus have a much bigger hit rate and score.

Table 6. SARI score by operation at turning points in Fig. 3

Control token	Value	SARI_add	SARI_keep	SARI_del	SARI
DTD_joint	0.2	2.71	27.03	69.32	33.02
	0.6	5.24	58.50	57.51	40.41
	1.0	3.30	62.64	26.68	30.87
	1.5	4.41	62.66	27.82	31.63
WR_joint	0.5	5.10	37.47	68.54	37.04
	0.75	6.65	54.91	62.57	41.37
	1.0	3.38	62.04	29.90	31.77
	1.25	4.19	54.88	58.35	39.14
LV_joint	0.2	7.15	50.83	63.83	40.60
	0.7	9.14	60.15	57.60	42.30
	1.0	2.25	61.62	32.17	32.01
LR_joint	0.2	1.80	19.27	69.46	30.18
	0.65	5.54	56.84	59.36	40.56
	1.0	2.43	62.42	15.26	26.70
	1.2	5.80	61.46	26.03	31.10

5.3. Performance of control token predictors

Table 7 shows the performance of regression and expectation and median of multi-classification predictors along with the average variance from the reference sentences. The expectation is the weight product of all predictions with probability, while the median is the prediction that splits the probability distribution into half. The value of the MAE and RMSE is low (0.063–0.149) because the unit distance between the two adjacent control tokens is 0.05.

Upon examining the performance of the predictors for different control tokens, notable variations can be observed. The DTD_predictor exhibits the highest R^2 score among all four predictors, indicating the strongest ability to capture the relationship between the value of the control token and the corresponding input sentences. Following closely is the LR_predictor, which also demonstrates a respectable R^2 score. In contrast, the LV_predictor in the regression method performs noticeably worse compared to its counterpart in the classification method. The R^2 scores are positive for the predictors of dependency tree depth ratio, replace-only Levenshtein ratio and length ratio in both methods, indicating their proficiency in capturing the desired relationship. However, the R^2 score is negative for the predictor of word rank ratio in both regression and classification methods, suggesting the failure of the proposed predictor to effectively capture the corresponding feature. Lastly, the performance of all classification predictors surpasses that of the regression predictors in the direct evaluation.

Figs. 4 and 5 show the density histogram and box plot of the predicted value, the mean and all values from reference sentences. Fig. 4a shows the histogram of all values in red dots, which exhibits a highly centralised impulse around the value of 1, with a noticeable gap on both sides. The remaining values predominantly occupy the left section of the distribution, ranging from 0.2 to 0.9. In contrast, the distribution of mean values, indicated by the green dashed line, demonstrates less centralisation and a closer approximation to a normal distribution. In Fig. 5a, similar to

Table 7. Performance of regressive control token predictors on the average value of ASSET test set (Average Variance means the average value of the variances calculated from different sentence pairs in ASSET test set)

Control token predictor	Strategy	MAE ↓	RMSE ↓	R^2 ↑	Average variance
DTD_predictor	regressive	0.117	0.149	0.508	0.045
	expectation	0.108	0.140	0.565	
	median	0.124	0.160	0.431	
WR_predictor	regressive	0.048	0.063	−0.204	0.003
	expectation	0.046	0.063	−0.198	
	median	0.046	0.063	−0.198	
LV_predictor	regressive	0.086	0.115	0.213	0.017
	expectation	0.071	0.094	0.470	
	median	0.093	0.116	0.198	
LR_predictor	regressive	0.098	0.125	0.527	0.033
	expectation	0.094	0.121	0.557	
	median	0.113	0.143	0.384	

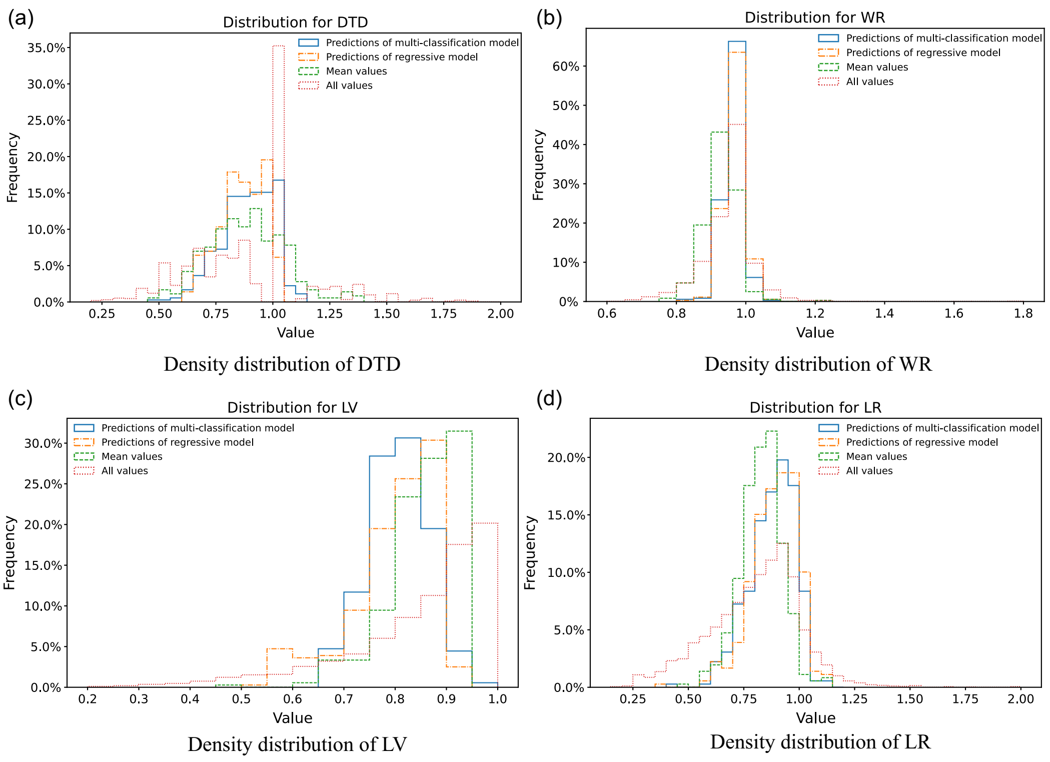

Figure 4. The density distribution of predictions, average values and values of all reference sentences.

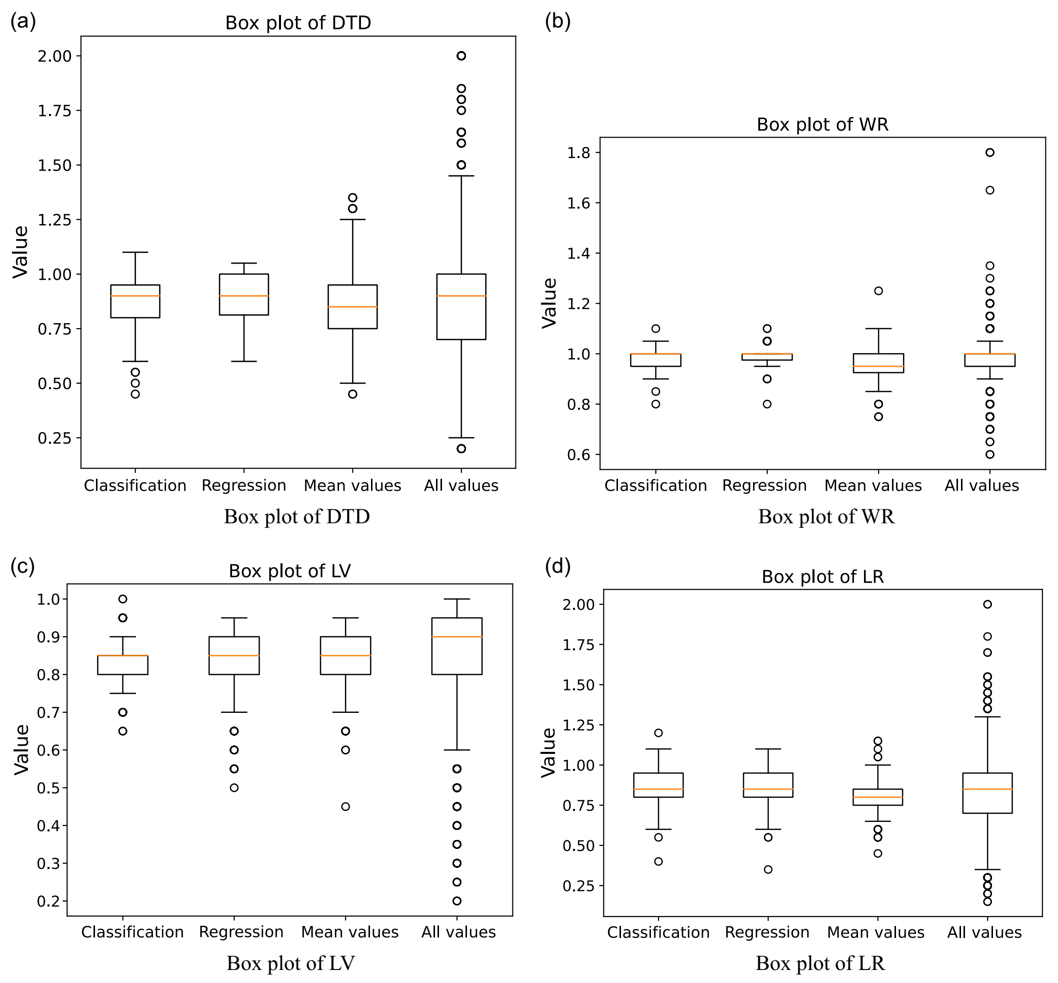

Figure 5. The box plot of distributions of predictions, average values and values of all reference sentences for the four control tokens.

Fig. 4a, all values have the most outliers and the largest range, while the predicted methods show higher concentration. The classification model overlaps more with mean values in the quartiles compared to the regression model.

Fig. 4b shows the histograms of all values in the red dots and the mean values in the green dashed line, which exhibit a similar shape, with the median values tending to centralise around 0.9. However, both the distribution of the regression model and the classification model appear more concentrated around the value of 0.95. These observations regarding the distribution align with the R^2 scores presented in Table 7, indicating that the predictors struggle to capture the relationship between input sentences and the word rank ratio when compared to the overall means. Similar to Fig. 4b, Fig. 5b shows that the box plot for all values overlaps more with mean values. However, there is a notable long tail due to outliers.

Fig. 4c shows the histogram of all values in red dots, which exhibits a characteristic long tail, while the regression model and mean values share a similar shape. However, considering the R^2 scores provided in Table 7, it becomes evident that the classification model aligns more closely with the mean value. This discrepancy may be attributed to the left section of the regression model,

Table 8. Performance of single-control token models with predictors on ASSET test (Regression and Classification methods refer to the situation that single-control models working with the control token predictors, average method means the control token value is the average value of reference sentences and optimization method is the same as shown in previous sections)

	DTD	WR	LV	LR
Regression				
SARI ↑	36.10	32.75	40.64	37.19
BERTScore ↑	0.870	0.878	0.821	0.868
Expectation				
SARI ↑	36.66	32.67	40.95	37.69
BERTScore ↑	0.863	0.880	0.830	0.864
Median				
SARI ↑	35.73	32.31	38.92	36.48
BERTScore ↑	0.868	0.883	0.855	0.866
Average				
SARI ↑	37.66	35.78	41.16	39.88
BERTScore ↑	0.859	0.861	0.837	0.849
Optimisation joint				
SARI ↑	40.71	41.42	42.86	40.64
BERTScore ↑	0.812	0.733	0.782	0.785

which registers values lower than 0.7. The observed differences in the histograms clarify the performance of the predictors and their relation to the LV control token. Consistent with Fig. 4c, Fig. 5c shows that the classification model has a more centralised distribution, closely overlapping with mean values. The regression model resembles the mean values, but the classification model performs better. There are numerous outliers in all values, indicating extreme data points.

Fig. 4d shows the histogram of all values in red dots, which exhibits a less centralised distribution in comparison to the other three histograms. However, despite this relatively dispersed nature, the mean value itself displays a more centralised shape compared to the predictors. Notably, the histogram of the classification model aligns closely with the mean value distribution, further reinforcing the consistency between these two distributions. This aligns with the distribution characteristics and the R^2 scores presented in Table 7. Fig. 5d presents box plots for control token LR. Although it is difficult to distinguish differences between the classification and regression models, the gap in performance compared to mean values is evident. The box plot for all values aligns with the histogram in Fig. 4, showcasing diversity and variance.

5.4. Performance with control token predictors

Table 8 shows the SARI and BERTScore of single-control token models with different methods. The performance gap in SARI and BERTScore among the regression, expectation and median models aligns with the gap in the direct numeric evaluation in Table 7, which demonstrates both

Table 9. Performance of control token model with predictors on ASSET test. The top three rows are predicted values, and the bottom three rows are calculated or optimised values

Models	SARI ↑	BERTScore ↑
Regression	42.30	0.833
Expectation	42.76	0.828
Median (Predicted)	40.74	**0.850**
Median (Calculated)	44.56	0.836
Average (Calculated)	44.55	0.816
Optimisation joint	**44.58**	0.794

Table 10. The performance with only one static value of the control token (The value is referred from Table 4)

Static control token	SARI ↑	BERTScore ↑
DTD fixed at 0.35	43.97	0.813
WR fixed at 0.7	43.62	0.809
LV fixed at 0.85	42.98	0.839
LR fixed at 0.9	41.30	0.846

the reliability of our chosen metric SARI and BERTScore and competence of mean value as an approximation to the ideal value. As a comparison, the scores of the average and median values calculated from the reference sentences and the optimised value found on the test set are added. Although the SARI scores of predictors and the average value are lower than the optimised ones, BERTScore remains higher. Among all the control tokens, the LV has the closest gap between the two types of methods.

Similar to Table 8, Table 9 shows the SARI and BERTScore with 4 predictors working cooperatively with the combined control token model. The performance difference still aligns with the previous table. However, the SARI score difference between the average value and optimization method is very low, while the average value maintains a higher BERTScore, which shows the average value of multi-reference can be a proper approximation of the ideal value for each sentence.

5.5. Performance of hybrid method

Considering the poor alignment and performance of single-control token models for some of the predictors, we conducted a hybrid method of both optimization and prediction methods in Tables 10 and 11. In Table 10, we replace the one predictor with the fixed value, referred from the optimization results in Table 4, and found the impact of different control token predictors. The DTD control token predictor has the most significant negative impact on the SARI score, while the LR has the least. By replacing the DTD predictor with a fixed value of 0.35, we increased the SARI score from 42.76 to 43.97.

In Table 11, we replace one of the optimised values with predicted values from the classification method and find the performance differences with control token predictors. Notably, the one with the DTD predictor still shows the largest drop in the SARI score and the one with the LR predictor outperforms the optimization method in both the SARI score and BERTScore.

Table 11. The performance with only one dynamic value of the control token and the static value for the remaining control tokens are DTD:0.35, WR:0.7, LV:0.85 and LR:0.9, respectively (The value is referred from Table 4)

Dynamic control token	SARI ↑	BERTScore ↑
DTD	42.75	**0.837**
WR	43.24	0.833
LV	44.16	0.816
LR	**44.61**	0.808

Table 12. Effect of varying Length ratio with the others remaining at 1.0

Source	Reflection nebulae are usually blue because the scattering is more efficient for blue light than red (this is the same scattering process that gives us blue skies and red sunsets)	SARI	BERTScore
LR_1.2	Reflection nebulae are usually blue because the scattering is more efficient for blue light than red (this is the same scattering process that gives us blue skies and red sunsets) and because the light reflects off of them	21.70	0.825
LR_1.0	Reflection nebulae are usually blue because the scattering is more efficient for blue light than red (this is the same scattering process that gives us blue skies and red sunsets)	19.03	0.937
LR_0.8	Reflection nebulae are usually blue because the scattering is more efficient for blue light than red (this is the same scattering process that gives us blue skies)	25.73	0.879
LR_0.6	Reflection nebulae are usually blue because the scattering is more efficient for blue light than red	41.71	0.880
LR_0.4	Reflection nebulae are usually blue because the scattering is more efficient	39.85	0.846
LR_0.2	Reflection nebulae are usually blue in colour	36.11	0.594

6. Case study

6.1. Effect and limitation of control tokens

In this section, we show the effect of each control token in examples. Table 12 lists the outputs of one sample sentence with the Length Ratio varying from 1.2 to 0.2 while the other 3 control tokens stay at 1. The difference between adjacent sentences is underlined. The design of this control token is about the sentence length. The output with LengthRatio_1.2 has the highest character length, while the output with LengthRatio_0.2 has the smallest, and there is no variation when the LengthRatio is set to 1. The effect of LengthRatio is significant and well-learned by the model. However, the desired meaning of control tokens in this table is varying the sentence length while maintaining the character, lexical complexity and syntactical complexity identical to the source sentence, which is impossible. Although the model is well affected by the control tokens, the influence of the control token is not absolute and there are still limitations since the control token is not the only constraint applied to the model.

In Table 13, we show a set of outputs of the same source sentence with different ReplaceOnlyLevenshtein ratios while the other 3 control tokens stay at 1. Since the ReplaceOnlyLevenshtein ratio can not be larger than 1, here lists the sentences with ReplaceOnlyLevenshtein ratio varying from 1 to 0.2. The design of this control token is about the similarity in character level between the input and output. In output with LV_1.0, the sentence is identical to the source sentence. In output with LV_0.8 and LV_0.6, the two sentences are

Table 13. Effect of varying ReplaceOnlyLevenshtein ratio with the others remaining at 1.0

Source	Moderate to severe damage extended up the Atlantic coastline and as far inland as West Virginia	SARI	BERTScore
LV_1.0	Moderate to severe damage extended up the Atlantic coastline and as far inland as West Virginia	20.29	0.959
LV_0.8	Moderate to severe damage happened along the Atlantic coast and as far inland as West Virginia	44.30	0.897
LV_0.6	Moderate to severe damage happened along the Atlantic coast and as far inland as West Virginia	44.30	0.897
LV_0.4	In West Virginia, the storm caused moderate to severe damage along the Atlantic coast and inland	44.33	0.527
LV_0.2	The National Hurricane Center (NHC) said that the storm was a “major hurricane” and not a tropical storm	26.40	0.058

Table 14. Effect of varying WordRank ratio and some other ratios with the others remaining at 1.0

Source	Shade sets the main plot of the novel in motion when he impetuously defies that law, and inadvertently initiates a chain of events that leads to the destruction of his colony’s home, forcing their premature migration, and his separation from them	SARI	BERTScore
WR_0.8	Shade sets the main plot of the novel in motion when he deliberately defies that law, and inadvertently initiates a chain of events that lead to the destruction of his colony’s home, forcing their premature migration, and his separation from them	24.58	0.862
WR_0.6	Shade sets the main plot of the novel in motion when he does not follow that law, and inadvertently initiates a chain of events that lead to the destruction of his colony’s home, forcing their premature migration, and his separation from them	25.95	0.828
WR_0.4	Shade sets the main plot of the novel in motion when he impetuously defies that law, and inadvertently initiates a chain of events that lead to the destruction of his colony’s home, forcing their premature migration, and his separation from them	19.54	0.850
WR_0.4 LV_0.8	Shade sets the main plot of the novel in motion when he does not follow the law, and inadvertently starts a chain of events that lead to the destruction of his colony’s home, forcing them to move away, and his separation from them	39.60	0.775
WR_0.2 LV_0.8	Shade sets the main plot of the novel in motion when he does not follow the law, and inadvertently starts a chain of events that lead to the destruction of his colony’s home, forcing their premature migration, and his separation from them	32.22	0.817

identical because the model can only generate meaningful sentences with correct grammar that approach the requirements of control tokens. The output for LV_0.2 shows the highest difference, but the meaning is far from the source sentence, which also shows the limitation of controllable TS with the control tokens that there is no guarantee of the factual reality.

In Table 14, we show 2 sets of outputs with different WordRank ratios along with some other ratios and the control tokens unmentioned remain at 1. The design of this control token is about the complexity of words used in the sentence. In the second and third rows, the model replaces the ‘inadvertently’ with ‘deliberately’ and ‘does not follow’. However, when WR is set to 0.4, the model generates identical outputs. In the fifth and sixth rows, we set the LV to 0.8, which allows

Table 15. Effect of varying DependencyTreeDepth ratio with the others remaining at 1.0

Source	The four canonical texts are the Gospel of Matthew, Gospel of Mark, Gospel of Luke and Gospel of John, probably written between AD 65 and 100 (see also the Gospel according to the Hebrews)	SARI	BERTScore
DTD_1.2	The four canonical texts are the Gospel of Matthew, Gospel of Mark and Gospel of Luke, probably written between AD 65 and AD 100 (see also the Gospel according to the Hebrews)	34.48	0.882
DTD_0.8	The four canonical texts are the Gospel of Matthew, Gospel of Mark and Gospel of Luke. They are probably written between AD 65 and 100 (see also the Gospel according to the Hebrews)	36.04	0.946
DTD_0.6	The four canonical texts are the Gospel of Matthew, Gospel of Mark and Gospel of Luke. The Gospel of John was probably written between AD 65 and 100 (see also the Gospel according to the Hebrews)	32.21	0.922
DTD_0.4	The four canonical texts are the Gospel of Matthew, Gospel of Mark and Gospel of Luke. The Gospel of John was probably written between AD 65 and 100 (see also the Gospel according to the Hebrews)	32.21	0.922

more variation in the output, and WR to 0.4 and 0.2. The model further replaces the ‘initiates’ with ‘starts’ and ‘migration’ with ‘move away’. Similarly, when WR set 0.2, the model recovers the ‘move away’ to ‘migration’. The examples in this table show the limitation of the interrelationship among all control tokens.

In Table 15, we show the effect of the DependencyTreeDepth ratio. The design of this control token is about the complexity of sentence structure, which normally splits long sentences into shorter sentences. The output sentence with DTD_1.2 is the same as the input sentence. When the DependencyTreeDepth ratio decreases to 0.8, the model splits the long sentence into two shorter sentences. With a lower DTD_0.6, the model further reduces the syntactical simplicity by changing the demonstrative pronoun ‘they’. However, when the DependencyTreeDepth ratio further decreases to 0.4, there is no difference in the output, due to the limitation of other control tokens. This example also shows the inconsistency of the effect of control tokens and the importance of the proper value of control tokens that can affect the sentence significantly.

Based on the observation and comparison among the examples in Tables 12–15, we can confirm that in the given examples, the control tokens work as designed. As shown in Table 14, the control tokens also affect one another. The interference among the control tokens made them work as a whole rather than four separate ones, causing difficulties in tweaking one of the control tokens to achieve the desired outputs. In addition, as shown in the bottom row of Table 13, the misuse of the control token may lead to false content and mislead the users.

6.2. Effect and limitation of optimisation and prediction methods

Tables 16–18 show several examples of simplifications produced by optimisation and prediction methods and the output with average value as the reference on the test set of the ASSET. All three outputs are from the same model with different values of control tokens listed in the second column.

In Table 16, the optimisation method shows a typical factual error because of the fixed values in control tokens. There is no relationship between ‘website’ and ‘magazine’ related to ‘taken over’ in the source sentence. However, due to the LR and LV ratios being fixed on the whole test set, the simplification from the optimisation method has to maintain a longer and more different sequence than the prediction method in this case, which makes it tend to generate extra content to fulfil the requirements. As a result, it generates false content and changes the meaning of the source sentence, which will mislead readers. This could be more common in larger and more diverse

Table 16. Examples of limitations of the optimisation method (changed the meaning in the third row)

Source	The name survives as a brand for a related spin-off digital television channel, digital radio station, and website which have survived the demise of the printed magazine			
Method	Control token	Output	SARI	BERTScore
Optimisation	DTD_0.35 WR_0.85 LV_0.8 LR_0.85	The name survives as a brand for a related digital television channel, digital radio station, and website which have not been taken over by the magazine	43.49	0.781
Prediction	DTD_0.6 WR_1.0 LV_0.8 LR_0.7	The name survives as a brand for a digital television channel, digital radio station, and website which have also survived	43.48	0.752
Average	DTD_0.65 WR_0.95 LV_0.9 LR_0.85	The name survives as a brand for a related digital television channel, digital radio station, and website which have survived without the magazine	41.37	0.779

Table 17. Examples with mispredicted values (missing 'or' clause in the third row)

Source	A few animals have a chromatic response, changing colour in changing environments, either seasonally (ermine, snowshoe hare) or far more rapidly with chromatophores in their integument (the cephalopod family)			
Method	Control token	Output	SARI	BERTScore
Optimisation	DTD_0.35 WR_0.85 LV_0.8 LR_0.85	A few animals have chromatic response. They change colour in changing environments, either seasonally (ermine, snowshoe hare) or far more rapidly (the cephalopod family)	30.19	0.724
Prediction	DTD_0.85 WR_0.95 LV_0.9 LR_0.6	A few animals have chromatic response. They change colour in changing environments, either seasonally (ermine, snowshoe hare)	36.61	0.621
Average	DTD_0.8 WR_0.8 LV_0.75 LR_0.8	A few animals have chromatic response, changing colour in changing environments, either seasonally (ermine, snowshoe hare) or more rapidly (the cephalopod family)	25.36	0.680

Table 18. Examples with properly predicted values

Source	Since the end of the 19th century Eschelbronn is well known for its furniture manufacturing industry			
Method	Control token	Output	SARI	BERTScore
Optimisation	DTD_0.35 WR_0.85 LV_0.8 LR_0.85	Since the end of the 19th century, Eschelbronn has become very famous for its furniture	43.86	0.822
Prediction	DTD_0.85 WR_1.05 LV_0.9 LR_0.8	Since the end of the 19th century Eschelbronn is famous for its furniture making	39.85	0.920
Average	DTD_0.85 WR_0.95 LV_0.9 LR_0.8	Since the end of the 19th century Eschelbronn is famous for its furniture making	39.85	0.920

datasets. Although the output of the prediction method also changed the original meaning, it just loses some meaning without adding false content. The output of the average value best preserves the meaning in this example.

In Table 17, we show the consequence of misprediction. The output of the optimisation method and the average value is quite similar except the length of the optimisation method is a bit longer.

In the prediction method, the output sentence is incomplete due to the lower length ratio compared to the average value. Although there is also a gap in the DTD ratio in optimisation and prediction methods, there seems to be no obvious change in the syntactical complexity, which is aligned with the limitations mentioned in previous sections.

In Table 18, we show an example of desired predictions. Although there are some small variances in the value of control tokens, the output of the prediction method is identical to the output of the average value, which is used as the reference sentence. However, the optimisation method changes the original meaning in the source sentence.

We have included SARI and BERTScore at the instance level for each row in Tables 16–18. It can be seen from the variation in these scores that there is sometimes a discrepancy between the accuracy of the simplified text and the value of the score. As such these scores are best used when aggregated over a large dataset, where instance-level variations have less effect.

7. Discussion and future directions

7.1. Effect of quantisation

One phenomenon found during the optimisation section in the original project is that the score of recommended optimisation is even lower than the default values of control tokens at 0.8. A hypothesis emerged that continuous optimisation is not an ideal option to maximise the score. As shown in the four rows in Table 3, the score in reimplementation is higher with equivalent values in the four control tokens. There could be several reasons: the algorithm is not working as expected or the optimisation budget is not large enough to find better optimisations. The default tokenization method in the MUSS project that breaks the control tokens into pieces brings more noise and probably lowers the performance. Apart from the verbosity in optimal values, the long tokenization of the control token is another concern of noisy input. Although the results above show signs of such problems, they may become more serious with the increasing length of control tokens, especially for short sentences. It would be wiser to limit the unnecessary noise in the input to a lower level.

7.2. Similarities and differences among control tokens

Fig. 3 and Table 6 expose the reason for variation with the control token and provide a good illustration of nature in each control token. In single-control tokens, the peak points mainly fall between 0.6 and 0.7, and the score decreases with the value deviating from the peak point. However, there are still some differences among the control tokens. In the *DependencyTreeDepth Ratio* and *Length Ratio*, the reduction is more dramatic than the other two. In both graphs, the SARI_add decreases with the value deviating from the peak point and increases slowly when the value is bigger than 1. The SARI_keep and SARI_del fluctuate in the form of two half-phase shifted sine functions and the maximum sum is found in between the peaks. The graph of the *WordRank Ratio* shows some diversity in both Figs. 3b and 7b among the tokenization methods. Although there is no explanation for the deviations, the deviations show the potential of combining different tokenization methods. When focusing on the main section from 0.5 to 1, the graph shows characteristics similar to the graphs in the previous two control tokens. As for the *ReplaceOnlyLevenshtein Ratio*, the slope is milder on the left side and it seems to have less effect on the SARI score. Unlike the other three control tokens, this control token can only indicate the intensity of change but not the direction of change. Although the combined effects are still under research, a more effective control token could be a better solution.

As for the optimal value, they are the optimal values within the budget. When rerunning the code, it is quite common to have a different set of optimal values. This is one limitation of the current SOTA system and we propose the predictor to improve this drawback. In addition, the optimal values for one control token and the combined control tokens are different. The correlation among the control tokens presumably causes this variation. If the four control tokens can be designed to work independently, the graph on a single-control token can be directly used to find the optimal value. However, the graph of combined control tokens is bound to have some distortions for now. Based on the detailed graph, it is also clear that the value of control tokens can significantly affect the performance of the models trained in this way and should be treated carefully.

7.3. Control token predictors

In Fig. 4a, the distribution of all values with a green dot line shows a very high centralisation at 1.0 and other values are decentralised. This attribute makes it difficult to be regarded as a regression task and hence has the highest MAE and RMSE in Table 9. In Table 7, the prediction of DTD is less applicable in the four control tokens. In addition, in Table 8, both DTD and WR prediction models show a much lower SARI score than the optimisation method, which indicates the average value may not be a proper approximation to the ideal values. In addition, the design of DTD and WR may hardly reflect the features of sentences and probably needs revisions.

In Table 8, we show that the SARI score of the prediction method and average value for control token DTD and WR is quite low compared to the optimisation method. However, the BERTScore is significantly higher, which is reasonable since the goal set in the optimisation method is to maximise the SARI score only. However, we would like to emphasise the limitations of the SARI score. A sentence with a higher SARI score is not necessarily more meaningful at the sentence level, because the SARI score is only related to the word level. A sentence full of similar add, keep and delete operations to the reference with complete unreadable order will still have a high SARI score. If the only goal is to chase the SARI score, the model may generate some meaningless content as shown in Table 16.

7.4. Hybrid method

In Tables 10 and 11, we tested the hybrid methods with either the prediction method or optimisation method to evaluate the effect of certain control token predictors. Both tables show poor performance with the DTD predictor and WR predictor. However, unlike the poor alignment of WR in Table 7, the DTD demonstrates good performance in R^2 score. The main reason for the poor performance with the DTD predictor may be that this control token is not well designed to reflect the attribute of sentences or it is not suitable for the model to learn. In addition, the optimisation method with the LR predictor outperforms the original optimisation method.

7.5. Future work

During the experiments, we found the following points for further exploration in subsequent studies:

- As the control tokens can be regarded as customised prompts to the models, the applicability of control tokens or similar ideas is not only limited to TS tasks. With the popularity of prompt tuning, similar ideas can be applied to other tasks which need control ability to the outputs. In addition, the metric-oriented control token might also help.
- As the tokenization strategy shows no statistically significant effect on the performance in the control token experiments, a further study on the longer prompts may be worth doing, especially in the prompt-driven generative models with templates or formats.
- The SARI score highly relies on the quality and diversity of the references and can reflect only partially on the simplicity of the output. This could be overcome with a new

reference-less metric to measure the readability in different audiences' perspectives or measure the similarity of the outputs with text genre targeting specific audience groups with large language models.

- As shown in Sections 5.4 and 5.5, both the predictors and the average values of DTD and WR improve the SARI score marginally, which might be because of the design of these two control tokens. If some of the control tokens cannot fulfil the design purpose, it might be worth building some new control tokens or fulfilling the designed goal differently.
- As shown in Section 6.1, there is interference among the control tokens, impairing the usability and practicality of control tokens. One alternative solution is to break the system into several subsystems, build a pipeline to normalise the sentence and divide and conquer the subtasks in steps.
- Last but not least, the predictor learns only from the WikiLarge dataset, and cannot serve the purpose of giving precise predictions based on the needs of different user groups. To better achieve the goal of customised simplification for different user groups, further improvement in the control token-predictor mechanism is needed.

7.6. Concluding remarks

In the investigation, we reimplemented an updated version of ACCESS with fewer resources and equivalent performance and uploaded it to Huggingface. We show the effects and interactions of control tokens with different values and their effectiveness and importance, which can be used to balance user intention and performance. We give some suggestions for improvements in quantisation, prove the insignificance of different tokenization strategies of control tokens in our settings and propose a generative method on the value of control tokens to improve the generalisation on different datasets. In the generative method, we test the performance of predictors of different control tokens, increase the whole system's performance with this newly proposed method and explore the potential of this prediction method. Although there remain limitations in both the design and prediction of control tokens, the prediction method can promote the popularity of control token methods. At last, we verified the insignificance in the performance for each single research question we set, but several small improvements together can achieve significant improvement.

Acknowledgement. Some sections of the content of this manuscript have been presented previously at the Text Simplification, Accessibility, and Readability (TSAR) workshop at the EMNLP2022 (Li, Shardlow, and Hassan 2022). This manuscript expands significantly upon the previously presented work.

References

Agrawal S., Xu W. and Carpuat M. (2021). A non-autoregressive edit-based approach to controllable text simplification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3757–3769.

Alva-Manchego F., Martin L., Bordes A., Scarton C., Sagot B. and Specia L. (2020a). ASSET: A dataset for tuning and evaluation of sentence simplification models with multiple rewriting transformations. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 4668–4679. Online.

Alva-Manchego F., Martin L., Scarton C. and Specia L. (2019). *EASSE: Easier automatic sentence simplification evaluation.* In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP): System Demonstrations*, Hong Kong, China. Association for Computational Linguistics, pp. 49–54.

Alva-Manchego F., Scarton C. and Specia L. (2020b). Data-driven sentence simplification: Survey and benchmark. *Computational Linguistics* **46**(1), 135–187.

Alva-Manchego F., Scarton C. and Specia L. (2021). The (un)suitability of automatic evaluation metrics for text simplification. *Computational Linguistics* **47**(4), 861–889.

De Belder J. and Moens M.-F. (2010). Text simplification for children. In *Proceedings of the SIGIR Workshop on Accessible Search Systems*. New York: ACM, pp. 19–26.

Devaraj A., Wallace B.C., Marshall I.J. and Li J.J. (2021). *Paragraph-level simplification of medical texts*. In *Proceedings of the Conference. Association for Computational Linguistics. North American Chapter. Meeting*. NIH Public Access, vol. **2021**, p. 4972.

Devlin J., Chang M.-W., Lee K. and Toutanova K. (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)*, Minneapolis, Minnesota. Association for Computational Linguistics, pp. 4171–4186.

Devlin S. (1998). The use of a psycholinguistic database in the simplification of text for aphasic readers. In *Linguistic Databases*.

Dong Y., Li Z., Rezagholizadeh M. and Cheung J.C.K. (2019). Editnts: An neural programmer-interpreter model for sentence simplification through explicit editing. arXiv preprint arXiv:1906.08104.

Dušek O. and Jurčíček F. (2016). Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. arXiv preprint arXiv:1606.05491.

Flesch R. (1948). A new readability yardstick. *Journal of Applied Psychology* **32**(3), 221–233.

Floridi L. and Chiriatti M. (2020). Gpt-3: Its nature, scope, limits, and consequences. *Minds and Machines* **30**(4), 681–694.

Guo H., Pasunuru R. and Bansal M. (2018). Dynamic multi-level multi-task learning for sentence simplification. CoRR, abs/1806.07304.

Hovy E. H. (1990). Pragmatics and natural language generation. *Artificial Intelligence* **43**(2), 153–197.

Lebret R., Grangier D. and Auli M. (2016). Neural text generation from structured data with application to the biography domain. arXiv preprint arXiv:1603.07771.

Levenshtein V.I., et al. (1966). Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet Physics Doklady*, vol. **10**. Soviet Union, pp. 707–710.

Lewis M., Liu Y., Goyal N., Ghazvininejad M., Mohamed A., Levy O., Stoyanov V. and Zettlemoyer L. (2020). *BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension*. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pp. 7871–7880. Online.

Li Z., Shardlow M. and Hassan S. (2022). *An investigation into the effect of control tokens on text simplification*. In *Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022)*, Abu Dhabi, United Arab Emirates (Virtual). Association for Computational Linguistics, pp. 154–165.

Lin C.-Y. (2004). *ROUGE: A package for automatic evaluation of summaries*. In *Text Summarization Branches Out*, Barcelona, Spain. Association for Computational Linguistics, pp. 74–81.

Martin L., de la Clergerie É., Sagot B. and Bordes A. (2020a). *Controllable sentence simplification*. In *Proceedings of the 12th Language Resources and Evaluation Conference*, Marseille, France. European Language Resources Association, pp. 4689–4698.

Martin L., Fan A., de la Clergerie É., Bordes A. and Sagot B. (2020b). Multilingual unsupervised sentence simplification. arXiv preprint arXiv:2005.00352.

Mei H., Bansal M. and Walter M.R. (2016). *Listen, attend, and walk: Neural mapping of navigational instructions to action sequences*. In *Thirtieth AAAI Conference on Artificial Intelligence*.

Nishihara D., Kajiwara T. and Arase Y. (2019). *Controllable text simplification with lexical constraint loss*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: Student Research Workshop*, Florence, Italy. Association for Computational Linguistics, pp. 260–266.

Omelianchuk K., Raheja V. and Skurzhanskyi O. (2021). Text simplification by tagging. arXiv preprint arXiv:2103.05070.

Papineni K., Roukos S., Ward T. and Zhu W.-J. (2002). *Bleu: A method for automatic evaluation of machine translation*. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics, pp. 311–318.

Petersen S. E. and Ostendorf M. (2007). *Text simplification for language learners: a corpus analysis*. In *Workshop on Speech and Language Technology in Education*. Citeseer.

Radford A., Wu J., Child R., Luan D., Amodei D., Sutskever I., et al. (2019). Language models are unsupervised multitask learners. *OpenAI Blog* **1**(8), 9.

Rapin J. and Teytaud O. (2018). Nevergrad - a gradient-free optimization platform. *Available at:* https://GitHub.com/FacebookResearch/Nevergrad.

Reiter E. and Dale R. (1997). Building applied natural language generation systems. *Natural Language Engineering* **3**(1), 57–87.

Scialom T., Martin L., Staiano J., de la Clergerie É.V. and Sagot B. (2021). Rethinking automatic evaluation in sentence simplification. arXiv preprint arXiv:2104.07560.

Shardlow M. (2014). A survey of automated text simplification. *International Journal of Advanced Computer Science and Applications* **4**(1), 58–70.

Sun R., Lin Z. and Wan X. (2020). *On the helpfulness of document context to sentence simplification*. In *Proceedings of the 28th International Conference on Computational Linguistics*, Barcelona, Spain (Online). International Committee on Computational Linguistics, pp. 1411–1423.

Surya S., Mishra A., Laha A., Jain P. and Sankaranarayanan K. (2019). *Unsupervised neural text simplification*. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy. Association for Computational Linguistics, pp. 2058–2068.

Vaswani A., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł. and Polosukhin I. (2017). Attention is all you need. In *Advances in Neural Information Processing Systems, 30*.

Wen T.-H., Gasic M., Mrksic N., Su P.-H., Vandyke D. and Young S. (2015). Semantically conditioned lstm-based natural language generation for spoken dialogue systems. arXiv preprint arXiv:1508.01745.

Wubben S., van den Bosch A. and Krahmer E. (2012). *Sentence simplification by monolingual machine translation*. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Jeju Island, Korea. Association for Computational Linguistics, pp.1015–1024.

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

Xu W., Napoles C., Pavlick E., Chen Q. and Callison-Burch C. (2016). Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics* **4**, 401–415.

Yang Z., Dai Z., Yang Y., Carbonell J., Salakhutdinov R.R. and Le Q.V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems, 32*.

Zhang T., Kishore V., Wu F., Weinberger K.Q. and Artzi Y. (2019). Bertscore: Evaluating text generation with bert. arXiv preprint arXiv:1904.

Zhang X. and Lapata M. (2017). Sentence simplification with deep reinforcement learning. arXiv preprint arXiv:1703.10931.

Zhao S., Meng R., He D., Saptono A. and Parmanto B. (2018). *Integrating transformer and paraphrase rules for sentence simplification*. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, Brussels, Belgium. Association for Computational Linguistics, pp. 3164–3173.

Appenidx A

A.1. Optimisation budget

Fig. A1 shows the SARI score during the optimisation procedure in 128 times attempts. The highest score appears at the35th attempt and four of the top-five scores appear within 64 times. Even though a higher SARI score can be between 65 and 128, there is only a small performance gap

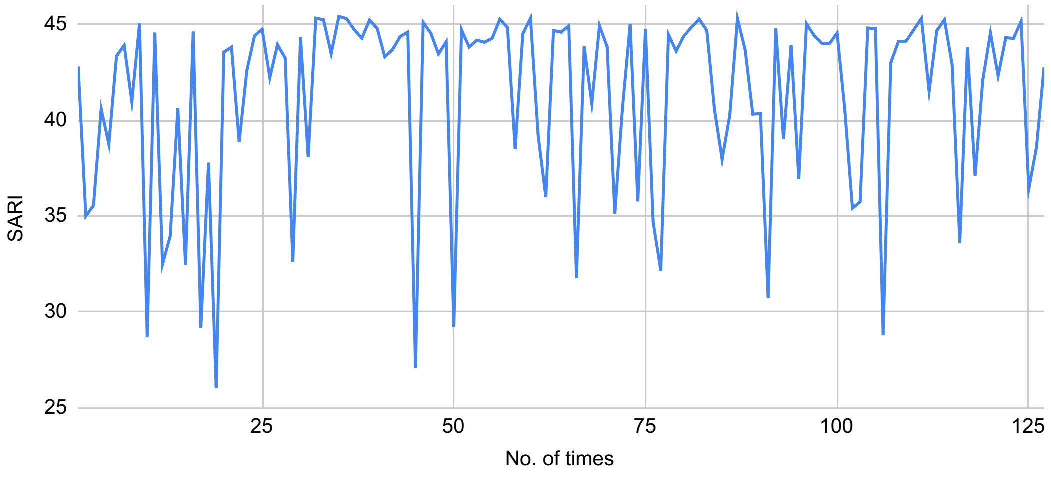

Figure A1. SARI score of 128 times of optimisation.

between the highest and the second-highest score. Thus it is reasonable to set the optimisation budget to 64.

A.2. Detailed singile control token results

Fig. A2 shows the BERTScore. Nearly all 3 tokenization strategies show high similarity to each other except Fig. A2(b). The figures show that nearly all models have the highest BERTScore around 1. Since the BERTScore calculates the correlation between the output and references, when the control token is set to 1, the model processes nothing, and the output can be quite similar to some of the references. Under this situation, as shown in Table 6, the SARI keep reaching the top. However, the peak of the BERTScore in Fig. A2(c) shifts to the left marginally, which shows that the references and input are not identical.

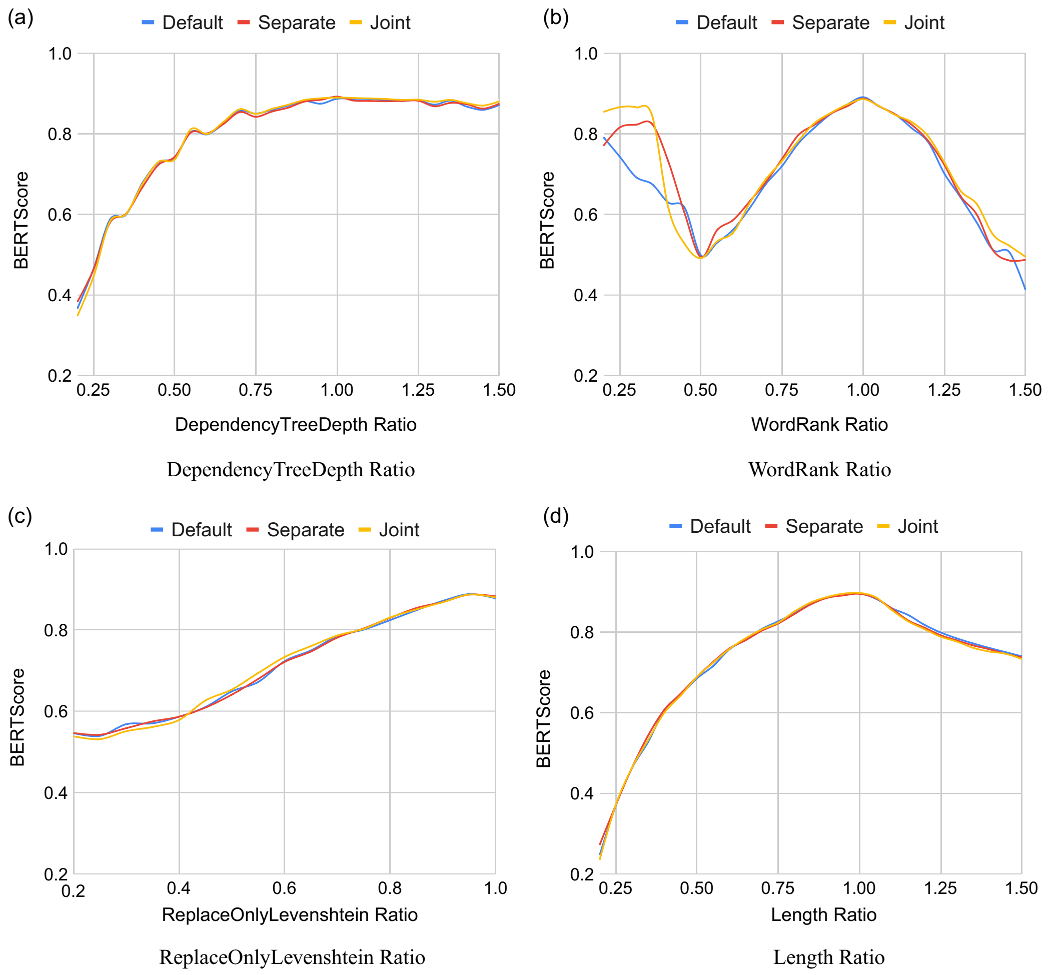

Figure A2. The effect of varying control tokens with different tokenization strategies on BERTScore.

A.3. More insights into the two outputs

We manually check the system outputs between the official system output of BART trained on the WikiLarge from MUSS and ours. In Table A1, we listed the findings and limitations of both models. The major differences include the following types:

- Shorter and separate sentences: We find many examples showing that our system prefers to split a long sentence into several shorter sentences. This may benefit aphasia people, who find it hard to follow very long sentences. On the other hand, it may also set obstacles to people having difficulties understanding the pronouns.
- Hallucination and false content: Both systems still generate hallucination and false content, which are inconsistent with the inputs. These false contents will undermine the practicality of the simplification systems. However, some of the false contents are caused by the control tokens themselves and can be alleviated by giving proper input value or a new control mechanism.
- Omitting contents: We also found that the two systems might have different preferences in deleted contents. For now, there lack of proper control mechanisms in control tokens to decide which part is the key information in the sentence and needs further improvement.

Table A1. More insights in the two systems

Insights	System outputs (MUSS followed by ours)
Our system's output tends to generate shorter sentences with demonstrative pronouns	They are castrated so that the animal may be more docile and put on weight more quickly They are castrated so that the animal may be more docile. It may put on weight more quickly
Both systems show hallucination and false content in some way. In the example, MUSS falsely predicted the meaning of 'It'	Stralsund is located on the coast of the Baltic Sea, near the city of Stralsund It is located on the Baltic Sea. The city of Stralsund is nearby
Both systems show hallucination and false content in some way. In the example, our system fails to simplify the 'extremely competitive' with 'important'	Admission to Tsinghua is very difficult Admission to Tsinghua is very important
In this example, our system shows a more readable output by sentence splitting and replacing 'Public Broadcasting Service' with 'television'	She performed for President Reagan in 1988's Great Performances at the White House series on the Public Broadcasting Service She performed for President Ronald Reagan in 1988's Great Performances at the White House series. The series was shown on television
Both system outputs omitted some information, 'motor racing championship' in MUSS and 'Brecia' in ours	The first Italian Grand Prix took place on September 4, 1921 at Brescia in Italy The first Italian Grand Prix motor racing championship was held on September 4, 1921