An Investigation into the Effect of Control Tokens on Text Simplification

Zihao LI **Matthew Shardlow** **Saeed-Ul Hassan**
Manchester Metropolitan University
21443696@stu.mmu.ac.uk {m.shardlow,s.ul-hassan}@mmu.ac.uk

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 ACCESS, (2) explored the effects of varying control tokens, (3) tested the influences of different tokenization strategies, and (4) demonstrated how separate control tokens affect performance. We show variations of performance in the four control tokens separately. We also uncover how the design of control tokens could influence the performance and propose some suggestions for designing control tokens, which also reaches into other controllable text generation tasks.

1 Introduction

Text simplification (TS) refers to reducing linguistic complexity at both syntactic and lexical levels without losing the main content (Alva-Manchego et al., 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 et al., 2015), for instance, depending on the characteristics of the dataset. The task can be roughly divided into sentence-level simplification (Nishihara et al., 2019; Martin et al., 2020a) and paragraph-level simplification (Sun et al., 2020; Devaraj et al., 2021). The two types of tasks may have different focuses, and this paper only involves sentence-level simplification.

In order to fit the requirements of different user groups, some projects introduced explicit discrete prompts as control tokens to assist the model in learning from datasets and adjusting the simplifications (Martin et al., 2020a; Agrawal et al., 2021).

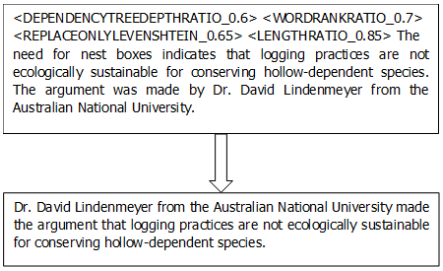


Figure 1: Example of input and output

By adjusting the value in different control tokens, researchers can manually adjust the characteristics of the output, such as length, syntactic and lexical difficulties, 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 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 Figure 1 and Appendix A. 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) Verify the importance of control tokens in Section 4.2. (2) Reimplement the ACCESS (Martin et al., 2020a) used in the current state-of-the-art (SOTA) in Section 3.3. (3) Explore the influence of the variation of control tokens in the format in Section 4.1. And finally (4) investigate the effects of the tokenization method in Section 4.2.

Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022), pages 154 - 165
December 8, 2022

2 Literature Review

Natural language generation (NLG) is a sub-task in natural language processing. There have been attempts to build an NLG system based on handcrafted 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 introduction of neural networks, more neural-network-based statistical methods were applied (Wen et al., 2015; Dušek and Jurčíček, 2016; Lebret et al., 2016; Mei et al., 2016). One important change happened with the publishing of the transformer architecture (Vaswani et al., 2017), which inspired the "pre-train and fine-tune" paradigm. Later, due to the new architecture outperforming existing ones 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, text simplification can also be regarded as monolingual machine translation (Wubben et al., 2012). With the development of sequence-to-sequence machine translation, text simplification also drew more attention (Guo et al., 2018; Surya et al., 2019; Omelianchuk et al., 2021)

In recent years, researchers tried to introduce 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 lay 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 lay 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 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 text simplification 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 multilingual tests were made in the MUSS, they were delivered separately and had little interference with the aim of this project. Thus, there is little need to focus on their research in French and Spanish.

The metrics also play a vital role in evaluating the performance of models. Although current metrics can hardly 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 (BLEU) (Papineni et al., 2002) and Recall-Oriented Understudy for Gisting Evaluation (ROUGE) (Lin, 2004) and non-reference-based metrics like Flesch-Kincaid Grade Level (FKGL) (Flesch, 1948). Currently, the most popular metric for text simplification is the system output against references and against the input sentence (SARI) (Xu et al., 2016). SARI is designed especially for text simplification 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 et al., 2021). As for the non-reference-based metrics, the BERT score is a BERT-based metric that evaluates the similarity between input and output 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.

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', '>', . . .]

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

3 Experiments

3.1 Quantisation differences

As mentioned in the literature review, there are 4 types of control tokens: <DEPENDENCYTREEDEPTH_*x*> (DTD), <WORDRANK_*x*> (WR), <REPLACEONLYLEVENSHTEIN_*x*> (LV) and <LENGTHRATIO_*x*> (LR). In the preprocessing step, they are calculated and added to the beginning of complex sentences in the complex dataset. As an augmentation to the control tokens, the calculated values are rounded to the nearest 0.05. However, in the original optimisation process, the calculated values by the algorithm provided by the Nevergrad (Rapin and Teytaud, 2018) API have high precision and verbose digits, just like the first line in Table 2. During the reimplementation, we found that only the first one or two digits are recognised as input values and the remaining digits didn't 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 ones like 0.2, 0.25, 0.3, ..., 1.0 and changed to the corresponding discrete algorithm in Nevergrad (Rapin and Teytaud, 2018).

3.2 Tokenization Strategies

One of the aims of this project is 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: One is to regard the whole control token as one token in the tokenizer; the other 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.

3.3 Reimplementation of ACCESS

One of the goals of this project is to reimplement and verify the effect of control tokens in the current SOTA. However, since the main focus of this project is on the control tokens, instead of training on both supervised and unsupervised datasets, it would be more practical to claim the reimplementation of ACCESS rather than MUSS. In order to build a unified baseline, this project also applied the BART model (Lewis et al., 2020), which is adopted in the MUSS project. The original project can be divided into the following sections: data mining, preprocessing, training, evaluation and optimisation.

Since the goal is verification, there is no need to rewrite the code for all sections. Thus only the codes related to training and some other peripheral functions have been altered to achieve similar results. The other functions, such as preprocessing and optimisation, still kept most of the original code. The original core API used for training is fairseq. This project replaced it with another open-source API — Huggingface. Huggingface provides a collection of the most popular pre-trained models and datasets, including the BART (Lewis et al., 2020) and a unified, advanced and user-friendly API to achieve the most common applications, which made 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 computer 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 the metrics, the SARI score is kept as the primary evaluation method (Xu et al., 2016), and the BERT score 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 strategies is corresponding to one model and there is a total of 16 models that need to be fine-tuned. This is why only BART-base is applied and the training epochs are limited. As for the reason for choosing 10 as the targeting epoch number, it 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. The results of the baseline shown in the next section can also partially prove the training process is probably long enough.

3.4 Training process

General NLP tasks can be divided into three steps: data preprocessing, training and evaluation. The preprocessing step followed the MUSS project (Martin et al., 2020b). In this project, there is one more step: optimisation. 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 <DEPENDENCYTREEDEPTH_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). 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; only the corresponding control tokens are kept in that dataset.

The next step is training. It follows the majority of fine-tuning processes for pretrained language models. By feeding the preprocessed complex-simple sentence pairs to the model, the model is expected to learn how to simplify texts and the meaning of each control token. As explained in the tokenization strategy, each tokenization method demands a separate model. To compare the performance of different tokenization methods, except the baseline, 15 models are fine-tuned in the experiment: 3 models with full control tokens and 12 models with only one control token. The models with one control token are used to verify the importance of combined control tokens and provide supportive evidence for the assumption.

The following step is evaluation. Thanks to Easier Automatic Sentence Simplification Evaluation(EASSE), 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 BERT score is added as a second reference. Different from the common applications in other projects, the BERT score 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-

Prompts	SARI	BERT	DTD	WR	LV	LR
Baseline	43.83	—	0.249...	0.814...	0.758...	0.858...
Default	44.00±0.05	0.754	0.25	0.8	0.75	0.85
Joint tokens	44.02±0.05	0.769	0.25	0.8	0.75	0.85
Separate tokens	44.04±0.05	0.754	0.25	0.8	0.75	0.85
Default	44.36±0.05	0.733	0.6	0.7	0.65	0.85
Joint tokens	44.58±0.05	0.794	0.35	0.85	0.8	0.85
Separate tokens	44.53±0.05	0.784	0.35	0.75	0.8	0.85
Default	43.34±0.06	0.827	0.6	0.85	0.85	0.85
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 2: Results on SARI and BERT score under differing tokenization strategies, with comparison to the baseline (top 4 rows of results), optimised parameter values (middle 3 rows) and values reported on unified parameters (last 3 rows).

simple pairs, and each complex sentence has ten reference simplifications.

The last 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 range. 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. With a budget of limitation to repeat the optimisation process 64 times, the algorithm can find a relatively optimised result. In order to ensure the reliability of the score under the optimised combination, a bootstrapping on the ASSET (Alva-Manchego et al., 2020a) test dataset will be executed by resampling the dataset 200 times and hence generate a 95% confidence interval.

4 Results

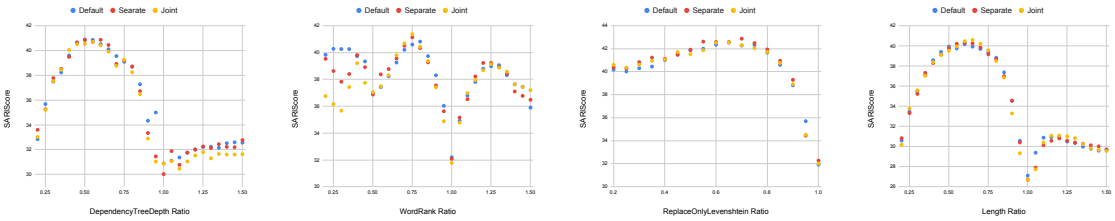
4.1 Overall performance

Following the setting in reimplementation, the baseline from the original code of the current SOTA is 43.83 on the ASSET (Alva-Manchego et al., 2020a) test dataset, which is consistent with the reported score in the MUSS in the corresponding scenario, which is 43.63±0.71. There is no confidence interval and BERT score 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 top 4 rows in the table2, the SARI score with 95% confidence in the reimplementation is slightly higher than the baseline. The middle 3 rows show the best SARI score with optimised options of control tokens. Among the 3 methods, the joint tokens had the highest SARI score. Interestingly, the BERT score is not always proportional to the SARI score, but the BERT score of optimal value is still quite high. The optimised values of control tokens are pretty close in all situations except the DTD. 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 worst. As for the BERT score, the joint tokenization method still outperforms the other two.

4.2 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 Figure 2. Except for the Figure 2(b), all 3 tokenization methods show a high consistency in the curves and have a common minimum at the value of 1. As shown in Table 4, it is mainly caused by the low score in both deletion and adding operations.

In addition to the curves, the differences in tokenization methods have marginal effects on the scores while the value of control tokens can change the performance significantly. In Figure 2(a) and



(a) DependencyTreeDepth Ratio (b) WordRank Ratio (c) ReplaceOnlyLevenshtein Ratio (d) Length Ratio

Figure 2: The effect of varying control tokens with different tokenization strategies on SARI Score.

Prompts	SARI	BERT	DTD
Default	40.82 ±0.05	0.805	0.55
	40.54 ±0.05	0.799	0.6
Separate	40.68±0.06	0.804	0.55
	40.87±0.05	0.801	0.6
Joint	40.71±0.06	0.812	0.55
	40.43±0.06	0.800	0.6

Prompts	SARI	BERT	WR
Default	40.61±0.06	0.720	0.75
	40.80±0.06	0.776	0.8
Separate	41.08±0.06	0.738	0.75
	40.32±0.05	0.797	0.8
Joint	**41.42±0.06**	0.733	0.75
	40.43±0.06	0.782	0.8

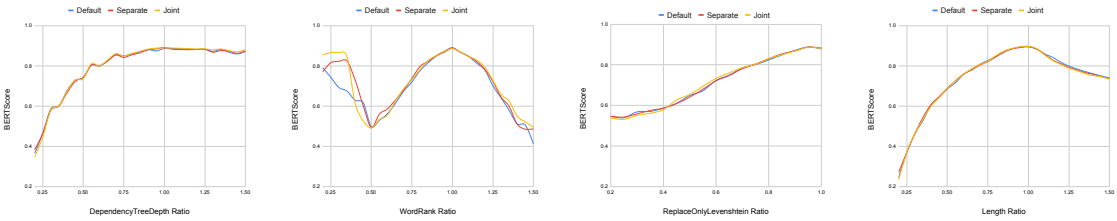
Prompts	SARI	BERT	LV
Default	42.52±0.06	0.750	0.65
	42.26±0.08	0.785	0.7
Separate	42.55±0.06	0.747	0.65
	42.86±0.06	0.782	0.7
Joint	42.63±0.06	0.761	0.65
	42.31±0.07	0.787	0.7

Prompts	SARI	BERT	LR
Default	40.15±0.06	0.758	0.6
	39.91±0.05	0.782	0.65
Separate	40.25±0.06	0.760	0.6
	40.27±0.05	0.781	0.55
Joint	40.46±0.05	0.758	0.6
	40.64±0.05	0.785	0.65

Table 3: Results on SARI and BERT scores of peak points in different control tokens.

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

Table 4: SARI score by operation at turning points in Figure 2.



(a) DependencyTreeDepth Ratio (b) WordRank Ratio (c) ReplaceOnlyLevenshtein Ratio (d) Length Ratio

Figure 3: The effect of varying control tokens with different tokenization strategies on BERT score.

2(c), the separate tokenization method shows the highest peak point, while in Figure 2(b) and Figure 2(d), the joint tokenization method has the best performance. The corresponding Table 3 also shows the scores in pairs under a unified value. Although the advantage is not as clear as the combined control tokens, the optimised SARI score of either separate or joint tokenization methods is still slightly higher than the default tokenization method.

The Table 4 is designed to help readers better understand the reason for variations in Figure 2. 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 much 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.

As for the BERT score, as shown in Figure 3, nearly all 3 tokenization strategies show high similarity to each other except Figure 3(b). The figures show that near all models have the highest BERT score around 1. Since the BERT score calculates the correlation between the output and references, when the control token is set to 1, the model processes nothing, and the output is very similar to the input. Under this situation, as shown in Table 4, the SARI_keep reaches the top. However, the peak of BERT score in 3(c) slightly deviates to the left, which shows that the references and input are not identical.

5 Discussion and Future Directions

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 first four rows in Table 2, the score in reimplementation is higher even in similar values. There are 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 shows sign of such problem, it may become more serious with the increasing of control tokens, especially for short sentences. It would be wiser to limit the unnecessary noise in the input to a lower level.

Figure 2 and Table 4 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 2. 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 2 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 Figure 2(b) and 3(b) 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 2 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 3 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, the most significant variation between single and combined control tokens is in DependencyTreeDepth Ratio. The optimal value in combined control tokens in the joint and separate tokenization method is 0.35 instead of 0.6. Although no direct comparison is listed in Table 2, comparing the middle and bottom three rows makes it pretty clear that 0.35 has a better SARI score. The correlation among the control tokens presumably causes this variation. There are also deviations in the other three control tokens. 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.

Another interesting finding between SARI and BERT in this paper is that most BERT score for optimal value is around 0.78 to 0.8. However, as shown in Figure 3(b) and 3(d), there are more than 1 points that have such value, so the BERT score alone cannot be used to evaluate the text simplification results. It may be a necessary but not sufficient condition for a good simplification. Since the SARI score is not perfect and relies on references, it is important to build non-reference-based metrics to evaluate the model on a different genre of corpora. The BERT score may play a role in these new metrics. Thus, this guess is worth further verification in future work.

In addition to the values, as shown in Table 3, the tokenization methods can also affect the peak score. In the curves, there are different optimised methods for each certain point. Although the performance differences may be caused by the fine-tuned models on a lower training scale, they may still imply performance variations between tokenization methods. Considering the various requirements of lay users, a mixed tokenization method based on the performance curve may maximise the model's performance at different points better than a fixed one. Although it remains unclear whether there will be the same effects in the combined control tokens, the mixed tokenizations method can be still promising with the appearance of more different control tokens. However, a more lightweight and efficient training method should be introduced to solve the problem of balancing cost and effect.

5.1 Future Work

In the future, one of the main tasks is to reimplement control tokens in different models or learning strategies so that training can be more lightweight and less time-consumed. Another goal is to build new non-reference-based metrics and replace SARI, which will significantly contribute to the development. However, it is not easy to understand the relationship between the performance and control tokens. A further investigation of the complex relationship between SARI and combined control tokens is also worth doing. Although the five-dimension graph may be less visualised, it can still provide some guidance on how to apply the control tokens. Designing and introducing new control tokens is another novel direction. The control tokens may be further simplified or optimised with a deeper inspection of the control tokens and SARI score. In addition to that, current optimisation procedure works only on the dataset level and needs more precise prediction on sentence level. A sentence level prediction model to the optimal value of control token may be worth considering. Lastly, whether there is a similar phenomenon of control tokens in other controllable text generation tasks is also an important question.

5.2 Concluding Remarks

In the investigation, we have shown the results and importance of control tokens with different values and tokenization methods, which can be used to balance user intention and performance. We proposed some improvements in quantisation, compared the influences of different tokenization strategies of control tokens and proposed possible further improvement means. Although the proposed suggestions may improve text simplification tasks marginally, they may also be generalised to prompts designing on other controllable NLP tasks.

References

Sweta Agrawal, Weijia Xu, and Marine Carpuat. 2021. A non-autoregressive edit-based approach to controllable text simplification. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 3757–3769.

Fernando Alva-Manchego, Louis Martin, Antoine Bordes, Carolina Scarton, Benoît Sagot, and Lucia Specia. 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*, pages 4668–4679, Online. Association for Computational Linguistics.

Fernando Alva-Manchego, Louis Martin, Carolina Scarton, and Lucia Specia. 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*, pages 49–54, Hong Kong, China. Association for Computational Linguistics.

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

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

Jan De Belder and Marie-Francine Moens. 2010. Text simplification for children. In *Prroceedings of the SIGIR workshop on accessible search systems*, pages 19–26. ACM; New York.

Ashwin Devaraj, Byron C Wallace, Iain J Marshall, and Junyi Jessy Li. 2021. Paragraph-level simplification of medical texts. In *Proceedings of the conference. Association for Computational Linguistics. North American Chapter. Meeting*, volume 2021, page 4972. NIH Public Access.

Ondřej Dušek and Filip Jurčíček. 2016. Sequence-to-sequence generation for spoken dialogue via deep syntax trees and strings. *arXiv preprint arXiv:1606.05491*.

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

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

Han Guo, Ramakanth Pasunuru, and Mohit Bansal. 2018. Dynamic multi-level multi-task learning for sentence simplification. *CoRR*, abs/1806.07304.

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

Rémi Lebret, David Grangier, and Michael Auli. 2016. Neural text generation from structured data with application to the biography domain. *arXiv preprint arXiv:1603.07771*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 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*, pages 7871–7880, Online. Association for Computational Linguistics.

Chin-Yew Lin. 2004. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

Louis Martin, Éric de la Clergerie, Benoît Sagot, and Antoine Bordes. 2020a. Controllable sentence simplification. In *Proceedings of the 12th Language Resources and Evaluation Conference*, pages 4689–4698, Marseille, France. European Language Resources Association.

Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2020b. Multilingual unsupervised sentence simplification. *arXiv preprint arXiv:2005.00352*.

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

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, Florence, Italy. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Vipul Raheja, and Oleksandr Skurzhanskyi. 2021. Text simplification by tagging. *arXiv preprint arXiv:2103.05070*.

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Sarah E Petersen and Mari Ostendorf. 2007. Text simplification for language learners: a corpus analysis. In *Workshop on speech and language technology in education*. Citeseer.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

J. Rapin and O. Teytaud. 2018. Nevergrad - A gradient-free optimization platform. https://GitHub.com/FacebookResearch/Nevergrad.

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

Thomas Scialom, Louis Martin, Jacopo Staiano, Éric Villemonte de la Clergerie, and Benoît Sagot. 2021. Rethinking automatic evaluation in sentence simplification. *arXiv preprint arXiv:2104.07560*.

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

Sai Surya, Abhijit Mishra, Anirban Laha, Parag Jain, and Karthik Sankaranarayanan. 2019. Unsupervised neural text simplification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 2058–2068, Florence, Italy. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Tsung-Hsien Wen, Milica Gasic, Nikola Mrksic, Pei-Hao Su, David Vandyke, and Steve Young. 2015. Semantically conditioned lstm-based natural language generation for spoken dialogue systems. *arXiv preprint arXiv:1508.01745*.

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, Jeju Island, Korea. Association for Computational Linguistics.

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

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

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *Advances in neural information processing systems*, 32.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Xingxing Zhang and Mirella Lapata. 2017. Sentence simplification with deep reinforcement learning. *arXiv preprint arXiv:1703.10931*.

Appendices

A

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).
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.
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).
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).
LR_0.6	Reflection nebulae are usually blue because the scattering is more efficient for blue light than red.
LR_0.4	Reflection nebulae are usually blue because the scattering is more efficient.
LR_0.2	Reflection nebulae are usually blue in color.

Table 5: Effect of varying Length ratio with the others remain 1.0.

Source	Moderate to severe damage extended up the Atlantic coastline and as far inland as West Virginia.
LV_0.8	Moderate to severe damage happened along the Atlantic coast and as far inland as West Virginia.
LV_0.6	Moderate to severe damage happened along the Atlantic coast and as far inland as West Virginia.
LV_0.4	In West Virginia, the storm caused moderate to severe damage along the Atlantic coast and inland.
LV_0.2	The National Hurricane Center (NHC) said that the storm was a "major hurricane" and not a tropical storm.

Table 6: Effect of varying ReplaceOnlyLevenshtein ratio with the others remain 1.0.

Source	He will abjure his allegiance to the king.
WR_0.8 LV_1.0	He will abjure his allegiance to the king.
WR_0.6 LV_1.0	He will abjure his allegiance to the king.
WR_0.8 LV_0.8	He will not give up his allegiance to the king.
WR_0.6 LV_0.8	He will not give up his power to the king.
WR_0.4 LV_0.8	He will not follow the orders of the king.
WR_0.2 LV_0.8	He will abjure his loyalty to the king.
WR_0.6 LV_0.8 LR_0.75	He will not follow the king anymore.

Table 7: Effect of varying WordRank ratio and some other ratios with the others remain 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).
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).
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).
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).
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).

Table 8: Effect of varying DependencyTreeDepth ratio with the others remain 1.0.