Controllable Lexical Simplification for English

Kim Cheng Sheang, Daniel Ferrés, Horacio Saggion
LaSTUS Lab, TALN Group, Universitat Pompeu Fabra
C/Roc Boronat 138, Barcelona, 08018, Spain
{kimcheng.sheang, daniel.ferres, horacio.saggion}@upf.edu

Abstract

Fine-tuning Transformer-based approaches have recently shown exciting results on sentence simplification task. However, so far, no research has applied similar approaches to the Lexical Simplification (LS) task. In this paper, we present ConLS, a Controllable Lexical Simplification system fine-tuned with T5 (a Transformer-based model pre-trained with a BERT-style approach and several other tasks). The evaluation results on three datasets (LexMTurk, BenchLS, and NNSeval) have shown that our model performs comparable to LSBert (the current state-of-the-art) and even outperforms it in some cases. We also conducted a detailed comparison on the effectiveness of control tokens to give a clear view of how each token contributes to the model.

1 Introduction

Lexical Simplification (LS) is a Natural Language Processing task that modifies texts by substituting difficult words with easier words (or phrases) while keeping the original information and meaning (Shardlow, 2014). Table 1 shows an example of a lexical simplification. On the other hand, Syntactic Simplification (SS) is a similar task that reduces the syntactic complexity of a text. Both LS and SS tasks can be seen as sub-tasks of the broader task of Automatic Text Simplification (Saggion, 2017), which reduces both the lexical and syntactic complexity of texts. Lexical Simplification systems (Paetzold and Specia, 2017a) usually have components for 1) identification of complex words; 2) generation of substitution words; 3) selection of the substitutes that can fit in the context; 4) ranking substitutes by their simplicity; and 5) morphological and contextual adaptation (if necessary). The systems evaluated in this paper do not perform complex word identification. We use datasets that already had a complex word tagged for each instance. Moreover, we do not address the morphological and context adaptation task because neural-based language models usually return a correct inflected candidate.

Complex Sentence:
The Hush Sound is currently on **hiatus**.
Simplified Sentence:
The Hush Sound is currently on **break**.

Table 1: A lexical simplification example taken from the LexMTurk dataset (Horn et al., 2014) with the complex word and the substitute word in bold.

The contributions of this paper are:

- To the best of our knowledge, we are the first to introduce a controllable mechanism for LS and to fine-tune a Transformer-based model for LS. [1]
- We have conducted an extensive evaluation of several metrics. This allows us to better understand the system when applied to real-world scenarios.

The rest of the paper is organized as follows: in Section 2, we describe related work on Lexical Simplification focusing on neural-based systems. Section 3 presents the ConLS approach. Section 4 describes the evaluation metrics and presents the experimental results. Section 5 discusses the results of the experiments, while Section 6 concludes the paper and presents future work.

2 Related Work

Early Lexical Simplification approaches with unsupervised models used: Latent Words Language Models (De Belder and Moens, 2010), Wikipedia-based models/rules (Biran et al., 2011; Yatskar et al., 2010; Horn et al., 2014) and distributional

[1] The code and data are available at https://github.com/KimChengSHEANG/ConLS

Proceedings of the Workshop on Text Simplification, Accessibility, and Readability (TSAR-2022), pages 199 - 206
December 8, 2022

lexical semantics (Glavaš and Štajner, 2015). (Paetzold and Specia, 2017b) started the use of neural networks for the task combined with a retrofitted context-aware word embedding model.

(Qiang et al., 2020, 2021) presented LSBert, a Lexical Simplification system that uses a pre-trained BERT (Devlin et al., 2019) model for English to generate substitution candidates. LSBert has two main phases: 1) Substitution Generation with the BERT Masked Language Model, and 2) Substitution Filtering and Ranking with several features: BERT prediction order, a BERT language model, PPDB database, corpus-based word frequency, and FastText similarity.

Martin et al. (2020) presented ACCESS a controllable Text Simplification system based on Sequence-to-Sequence models. This system allows explicit control of simplification conditions such as length, amount of paraphrasing, lexical complexity, and syntactic complexity. ACCESS achieved SOTA results in Text Simplification benchmarks on the WikiLarge test set. Later on, Martin et al. (2022) introduced MUSS (an extended version) by fine-tuning BART (Lewis et al., 2019) with ACCESS, and the results were improved. In addition, Sheang and Saggion (2021) took a similar approach, adding another control token (number of words) and fine-tuning it with T5 (Raffel et al., 2020).

3 System Description

Following recent works of Martin et al. (2020), Martin et al. (2022), Sheang and Saggion (2021), and Štajner et al. (2022b), we are inspired to apply a similar approach in lexical simplification task. Specifically, our model is based on Sheang and Saggion (2021), a model originally developed for sentence simplification[2]. We propose a controllable mechanism for LS because we believe that the embedded token values extracted from training data could give additional information to the model about the relations between the source and the target word; so that at inference, we could define different token values that fulfill our objectives, which in this case is to find the best candidates. In the following paragraphs, we describe all the details about each token and the reason why they are chosen.

[2] https://github.com/KimChengSHEANG/TS_T5

Word Length (WL) is the character length ratio between the complex word and the target word. It is the number of characters of the target word divided by the number of characters of the complex word. Based on our analysis of the training dataset (TSAR-EN), 65.71% of the time complex word is longer than the best candidate, 21.30% the complex word is shorter than the best candidate, and 12.99% both are the same length.

Word Rank (WR) is the inverse frequency of the target word divided by that of the complex word. The inverse frequency order is extracted from the FastText pre-trained model. Based on our analysis of the TSAR-EN dataset, 85.45% of the time, the complex word has a lower frequency than the best candidate. Therefore, this token is a good indicator to help guide the model to predict simpler candidates.

Candidate Ranking (CR) is the ranking order extracted from the training data. The values are given to candidates by the ranking order. E.g., the best-ranking candidate is given the value 1.00, the second 0.75, the third 0.50, the fourth 0.25, and starting from the fifth, it is given 0.00. We used only five different values to avoid overloading the model, as the training data is relatively small. In addition, the rationale behind using these values is that we want the model to learn candidates ranking from data through the training process rather than injecting additional information or doing post-processing.

4 Experiments

In our experiments, we compare our model with the current state-of-the-art model LSBert (Qiang et al., 2020). We used the original LSBert configurations and resources, and we made the following changes to have a detailed comparison with our model. By default, LSBert returns only a single best candidate for each complex word, so we made the changes to return the 10 best-ranked candidates. We changed the number of BERT mask selections from 10 to 15 so that after removing duplicate candidates, we still have around 10 candidates. Moreover, we filtered out all the candidates that were equal to the complex word. Due to the fact that all the used datasets have gold annotated simpler substitutions in all instances, we could assume that returning the complex word would be incorrect.

4.1 Datasets

This subsection describes all the Lexical Simplification datasets for English that we used in our experiments. We used LexMTurk (Horn et al., 2014), BenchLS[3] (Paetzold and Specia, 2016a), and NNSeval [4] (Paetzold and Specia, 2016b) for testing and TSAR-EN (Štajner et al., 2022a) dataset for training and validation. LexMTurk has 500 sentences that were obtained from Wikipedia. This dataset contains the marked complex words and their replacements suggested by 50 English-speaking annotators. The BenchLS dataset is a union of the LSeval (De Belder and Moens, 2012) and LexMTurk datasets in which spelling and inflection errors were automatically corrected. The NNSeval dataset is a filtered version of the BenchLS adapted to evaluate LS for non-native English speakers.

Sentence
European Union foreign ministers agreed Monday to impose fresh sanctions on Syria as a U.N.-backed peace plan – along with all other diplomatic efforts – has yet to stop the **carnage** that mounts every day.
Simpler Substitutes
destruction:6, bloodshed:3, massacre:3, slaughter:3, carnage:2, brutality:1, butchering:1, butchery:1, damage:1, death:1, slaying:1, violence:1, war:1

Table 2: An example taken from the TSAR-EN dataset Štajner et al. (2012) with the target word in bold. The numbers after ':' represents the number of workers that suggested the substitution. Each instance has 25 substitutes suggested by 25 crowd-sourced workers.

TSAR-EN dataset has 386 instances with 25 gold-annotated substitutions. Table 2 shows an example. The instances and their target complex words were extracted from the Complex Word Identification shared task 2018 (Yimam et al., 2018). The instances were annotated using Amazon's Mechanical Turk by 25 annotators. A native English annotator reviewed all suggestions.

4.2 Evaluation Metrics

We evaluated the systems with several metrics that could take into account the results for different numbers of K candidates (from 1 up to 10). The metrics used are the following:

- *Accuracy@1*: is the ratio of instances with the top-ranked candidate in the gold standard list of annotated candidates.
- *Accuracy@K@top1*: The ratio of instances where at least one of the top K predicted candidates matches the most frequently suggested synonym/s[5] in the gold list of annotated candidates.
- *Potential@K*: the percentage of instances for which at least one of the top K substitutes predicted is present in the set of gold annotations.
- *Mean Average Precision@K (MAP@K)*: This metric evaluates the relevance and ranking of the top K predicted substitutes.
- Precision@K: the percentage of top K generated candidates that are in the gold standard.
- Recall@K: the percentage of gold-standard substitutions that are included in the top K generated substitutions.

4.3 Experimental Setup

In this section, we describe how the data are preprocessed, the training details of the model, and finally, the generation of candidates.

4.3.1 Data Preprocessing

For each instance, we have a sentence, a complex word, and a list of ranked candidates. We compute all the ratios and the ranking, then prepend it to the source sentence. We also use special tokens [T] and [/T] to mark the boundary of the complex word in the source sentence and the simple word in the target sentence. Moreover, these special tokens help us identify the candidates during the inference. Table 3 shows an example of source and target sentences embedded with token values and boundary tokens.

4.3.2 Training

For our experiments, we fine-tuned T5-Large on the TSAR-EN dataset. We also compared the differences of T5 models; the results are in Table 6. We split the dataset to 90% for training and 10% for validation. This 10% validation set is also used

[3] https://doi.org/10.5281/zenodo.2552393
[4] https://doi.org/10.5281/zenodo.2552381
[5] Ties in the most repeated gold-annotated candidates are taken into account.

Source: <CR_1.00> <WL_0.54> <WR_0.90> The Obama administration has seen what The New York Times calls an [T]unprecedented[/T] crackdown on leaks of government secrets.
Target: The Obama administration has seen what The New York Times calls an [T]unusual[/T] crackdown on leaks of government secrets.

Table 3: A training example. The control token values are extracted from the complex word (unprecedented) and one substitute word (unusual). The word unusual is the best-ranked candidate suggested by annotators, so the CR value is 1.00. We used all the candidates in each instance to generate parallel sentences for training. One candidate per training example.

in the token values search at the inference, as described in the following section. For the training data, we preprocessed by extracting and adding control tokens to the source sentence along with the boundary tokens to the complex word and substitute word, as shown in Table 3. We set the maximum sequence length (number of tokens) to 128, as all our datasets contain less than 128 in tokens length. We used Optuna (Akiba et al., 2019) for hyper-parameters search. For more details about the implementation and hyperparameters, please check Appendix A.

4.3.3 Inference

First, we performed token values search on the validation set that maximizes the Accuracy@1@top1 score using Optuna (Akiba et al., 2019). We searched the values ranging between 0.5 and 1.25; at each iteration, we changed the value by 0.05. We searched only WL and WR, whereas for CR, we set it to 1.00 because we already knew that the best-ranking candidates were given the value of 1.00. Then we kept these values fixed for all sentences at the inference. Finally, at the inference, we set the beam search to 15 and the number of return sequences to 15 so that after filtering out some duplicate candidates, the remaining would be around 10. The ranking order of the candidates is chosen from the return orders of sequences produced by the model.

5 Results and Discussion

In Table 4 we present the results for the metrics: *Accuracy@1*, *Accuracy@k@Top1*, and *Potential@K*. In Table 5 we present the results for the metrics: *MAP@K*, *Precision@K*, and *Recall@K*. The results of ConLS presented here are based T5-Large.

Our experiments show that the modified LSBert had improved its *Accuracy@1* metric results with respect to the ones seen in the original LSBert paper (Qiang et al., 2021): *Accuracy@1* has improved from 79.20 to 84.80 for LexMTurk, from 61.60 to 67.59 for BenchLS, and from 43.60 to 44.76 for NNSeval. On the other hand, for the *Accuracy@1* metric the ConLS system does not improve the results of the modified LSBert system but improves the results of the original LSBert for the LexMturk and BenchLS datasets. The results of the *Accuracy@k@Top1* metric show that the modified LSBert achieves better results at K={1,2} and the ConLS achieves better results at K={3,4,5} for all datasets. This indicates that with more candidates allowed (3, 4, and 5 candidates) the ConLS is able to generate more instances with candidates within the top-1(s) gold annotated substitution(s) with respect to LSBert. The results of the *Potential@K* metric show these facts: 1) in LexMturk and BenchLS, the ConLS is outperforming LSBert gradually and increasingly from k=3 to k=10; 2) in NNSeval, ConLS improves the potential of LSBert only at K=10. For the MAP@K metric, we show that ConLS is able to improve the results of the metric at K={4,5,10} in all the datasets with respect to the modified LSBert. Finally, the results of the *Precision@K* and *Recall@K* metrics show the same pattern: 1) for LexMTurk, ConLS outperforms the LSBert in all K={3,5,10}; 2) for BenchLS and NNSEval, ConLS outperforms the LSBert only in K={5,10}.

We also conducted a comparison on the effect of different T5 models trained with TSAR-EN and evaluated with LexMTurk. Table 6 shows that the T5-Large model performs a lot better than the T5-Base and the T5-Small models in all metrics (Accuracy@1, Accuracy@k@Top1). Therefore, we believe that the performance of our model would improve if we could go with larger model, for example, T5-3b or T5-11b. We have tried with T5-3b model, but unfortunately it was unable to fit into our GPU memory (NVidia RTX 3090) even though we had set the batch size to as small as one.

To evaluate the effectiveness of the control tokens, we conducted further experiments with different set of combinations. We trained and evaluated each set of tokens using T5-Large with TSAR-EN

Dataset	System	ACC@1	ACC@k@Top1					Potential@k				
			@1	@2	@3	@4	@5	@2	@3	@4	@5	@10
BenchLS	LSBert	**67.59**	**40.68**	**51.45**	57.37	59.84	61.57	**77.07**	**81.27**	83.32	84.28	85.47
	ConLS	62.00	37.99	51.34	**59.31**	**64.90**	**68.46**	74.92	**81.27**	**84.82**	**87.08**	**90.31**
NNSeval	LSBert	**44.76**	**28.03**	**38.49**	43.93	46.86	49.79	**59.00**	**64.85**	**67.78**	**71.55**	74.48
	ConLS	41.00	26.77	34.30	**45.18**	**50.20**	**52.71**	53.14	61.09	65.69	69.87	**79.08**
LexMTurk	LSBert	**84.80**	**44.00**	54.80	60.40	61.80	62.80	**91.00**	93.20	94.60	95.00	95.80
	ConLS	80.60	43.80	**56.39**	**65.40**	**71.20**	**76.60**	90.00	**95.60**	**97.40**	**98.20**	**99.60**

Table 4: The results of LSBert and ConLS for the metrics: Accuracy@1, Accuracy@k@Top1, and Potential@K.

Dataset	System	MAP@k					Precision@k			Recall@k		
		@2	@3	@4	@5	@10	@3	@5	@10	@3	@5	@10
BenchLS	LSBert	**52.26**	**42.29**	34.79	29.25	15.74	**46.46**	34.62	24.90	**25.74**	29.80	32.41
	ConLS	49.73	41.37	**35.01**	**30.54**	**18.84**	46.34	**37.11**	**26.20**	25.59	**32.25**	**41.89**
NNSeval	LSBert	**34.93**	**27.84**	23.18	19.97	10.73	32.84	26.16	18.78	**19.55**	23.40	26.14
	ConLS	31.69	27.31	**23.23**	**20.30**	**12.53**	**32.91**	**27.02**	**19.51**	18.80	**23.80**	**32.08**
LexMTurk	LSBert	**67.05**	54.41	45.83	39.01	21.29	58.03	45.25	33.43	20.52	24.61	27.52
	ConLS	65.45	**55.45**	**48.04**	**42.52**	**27.59**	**60.16**	**49.89**	**36.94**	**21.32**	**27.51**	**37.15**

Table 5: The results of LSBert and ConLS for the metrics: *MAP@K*, *Precision@K*, and *Recall@K*.

T5 Model	ACC@1	ACC@k@Top1		
		@1	@2	@3
T5-Small	23.40	7.80	11.80	15.40
T5-Base	60.00	28.80	40.40	48.40
T5-Large	**80.60**	**43.80**	**56.39**	**65.40**

Table 6: The results of ConLS trained all tokens using different T5 models. The models were trained with TSAR-EN and evaluated with LexMTurk.

Tokens	ACC@1	ACC@k@Top1		
		@1	@2	@3
No Tokens	79.20	41.80	55.20	62.60
CR	79.00	41.00	54.40	62.60
WL	79.40	43.00	55.20	65.00
WR	78.60	41.20	54.60	63.20
CR+WL	78.40	41.40	54.40	62.40
CR+WR	78.60	42.80	54.60	62.20
WL+WR	78.60	41.00	54.20	62.20
All Tokens	**80.60**	**43.80**	**56.39**	**65.40**

Table 7: The results of ConLS trained with different set of tokens. Each model was trained with TSAR-EN and evaluated with LexMTurk.

for training and LexMTurk for evaluation. The results on Table 7 have shown that the model trained with no tokens performs lower than the model with all tokens in all metrics, especially for the Accuracy@1@Top1 metric, the model with all tokens perform +2 points higher. Moreover, the **all tokens** model performs better than all other models in all metrics. This indicates that each token contributes to the selection and the ranking of the candidates that leads to better performance.

6 Conclusions and Future Work

This paper presents ConLS, the first approach for Controllable Lexical Simplification. The paper also describes the evaluation of LSBert and ConLS for English with the LexMTurk, BenchLS, and NNSeval datasets for testing and the TSAR-EN dataset for training. The results of our evaluation show that the modified LSBert improves the *Accuracy@1* metric results with respect to the ones seen in the original LSBert paper in all three datasets. ConLS also improves it for the LexMturk and BenchLS datasets. Moreover, the ConLS system is able to achieve: 1) more potential to capture correct answers at K={3,4,5,10} for BenchLS and LexMturk and at K=10 for NNSeval with respect to LSBert, 2) with more candidates retrieved (4 or 5) is able to generate more candidates within the top-1 more frequent gold-annotated suggestions with respect to LSBert, 3) with K={5,10} candidates is able to generate (according to the gold-annotations) more correct and different candidates.

For future work, we plan to build a custom model to predict the best control token values from a given input instance. Having instance-customized control token values seems more adequate, as humans usually select the best candidate based on context.

Limitations

We describe in this Section the limitation of our work. The most probable limiting features are:

- The size of training dataset: the TSAR-EN dataset has 386 instances. Obviously, training with datasets with a large number of instances would be recommended to create better models.
- Quality of the training dataset: although during the creation of the TSAR-EN dataset, it was inspected and the unsuitable substitutions were removed and replaced with suitable ones (Štajner et al., 2022a), it is possible that the dataset quality could be improved by including substitutions not reported by the annotators.
- Quality of the testing datasets: it is also possible that these datasets could be improved by including substitutions not reported by the annotators.
- Successful adaptation to other languages: we could have possible difficulties in achieving similar adaptations and results in non-English languages due to the difficulties in availability of similar resources for other languages and specifically for low-resource languages.

Ethics Statement

We have described the limitations of the proposed method in the previous Section. All the scientific datasets and algorithms used are properly cited.

Acknowledgements

Our work is supported from the project Context-aware Multilingual Text Simplification (ConMuTeS) PID2019-109066GB-I00/AEI/10.13039/501100011033 awarded by Ministerio de Ciencia, Innovación y Universidades (MCIU) and by Agencia Estatal de Investigación (AEI) of Spain. In addition, we would like to thank the anonymous reviewers for their constructive comments and suggestions.

References

Takuya Akiba, Shotaro Sano, Toshihiko Yanase, T. Ohta, and Masanori Koyama. 2019. Optuna: A next-generation hyperparameter optimization framework. *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*.

Or Biran, Samuel Brody, and Noémie Elhadad. 2011. Putting it simply: a context-aware approach to lexical simplification. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 496–501, Portland, Oregon, USA. Association for Computational Linguistics.

Jan De Belder and Marie-Francine Moens. 2010. Text Simplification for Children. In *Proceedings of the SIGIR Workshop on Accessible Search Systems*, pages 19–26.

Jan De Belder and Marie-Francine Moens. 2012. A Dataset for the Evaluation of Lexical Simplification. In *Proceedings of the 13th International Conference on Computational Linguistics and Intelligent Text Processing - Volume Part II*, CICLing'12, page 426–437, Berlin, Heidelberg. Springer-Verlag.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186.

Goran Glavaš and Sanja Štajner. 2015. Simplifying Lexical Simplification: Do We Need Simplified Corpora? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 63–68, Beijing, China. Association for Computational Linguistics.

Colby Horn, Cathryn Manduca, and David Kauchak. 2014. Learning a Lexical Simplifier Using Wikipedia. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 458–463, Baltimore, Maryland. Association for Computational Linguistics.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2019. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461.

Ilya Loshchilov and Frank Hutter. 2019. Decoupled weight decay regularization. In *International Conference on Learning Representations*.

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

Louis Martin, Angela Fan, Éric de la Clergerie, Antoine Bordes, and Benoît Sagot. 2022. MUSS: Multilingual unsupervised sentence simplification by mining paraphrases. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 1651–1664, Marseille, France. European Language Resources Association.

Gustavo Paetzold and Lucia Specia. 2016a. Benchmarking Lexical Simplification Systems. In *Proceedings of LREC-2016*.

Gustavo Paetzold and Lucia Specia. 2016b. Unsupervised Lexical Simplification for Non-Native Speakers. *Proceedings of the AAAI Conference on Artificial Intelligence*, 30(1).

Gustavo Paetzold and Lucia Specia. 2017a. A Survey on Lexical Simplification. *Journal of Artificial Intelligence Research*, 60:549–593.

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

Jipeng Qiang, Yun Li, Zhu Yi, Yunhao Yuan, and Xindong Wu. 2020. Lexical simplification with pretrained encoders. *Thirty-Fourth AAAI Conference on Artificial Intelligence*, pages 8649—-8656.

Jipeng Qiang, Yun Li, Yi Zhu, Yunhao Yuan, Yang Shi, and Xindong Wu. 2021. Lsbert: Lexical simplification based on bert. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 29:3064–3076.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Horacio Saggion. 2017. *Automatic Text Simplification*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

Matthew Shardlow. 2014. A Survey of Automated Text Simplification. *International Journal of Advanced Computer Science and Applications*, 4.

Kim Cheng Sheang and Horacio Saggion. 2021. Controllable sentence simplification with a unified text-to-text transfer transformer. In *Proceedings of the 14th International Conference on Natural Language Generation*, pages 341–352, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45.

Mark Yatskar, Bo Pang, Cristian Danescu-Niculescu-Mizil, and Lillian Lee. 2010. For the sake of simplicity: Unsupervised extraction of lexical simplifications from Wikipedia. In *Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 365–368, Los Angeles, California. Association for Computational Linguistics.

Seid Muhie Yimam, Chris Biemann, Shervin Malmasi, Gustavo H. Paetzold, Lucia Specia, Sanja Stajner, Anaïs Tack, and Marcos Zampieri. 2018. A Report on the Complex Word Identification Shared Task 2018. *CoRR*, abs/1804.09132.

Sanja Štajner, Richard Evans, Constantin Orasan, and Ruslan Mitkov. 2012. What can readability measures really tell us about text complexity. In *Proceedings of workshop on natural language processing for improving textual accessibility*, pages 14–22. Citeseer.

Sanja Štajner, Daniel Ferrés, Matthew Shardlow, Kai North, Marcos Zampieri, and Horacio Saggion. 2022a. Lexical Simplification Benchmarks for English, Portuguese, and Spanish. *Frontiers in Artificial Intelligence*, 5.

Sanja Štajner, Kim Cheng Sheang, and Horacio Saggion. 2022b. Sentence simplification capabilities of transfer-based models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(11):12172–12180.

A Implementation Details

Our implementation is based on Huggingface Transformers (Wolf et al., 2020) and Pytorch-lightning[6]. We trained the model using T5-Large for 8 epochs. For the optimization, we used AdamW (Loshchilov and Hutter, 2019) optimizer with the learning rate of 1e-5 and adam epsilon of 1e-8. We set the batch size of 8 for both training and testing. For the inference, we used beam search with the size of 15 to get around 10 candidates after filtering out duplicate candidates or the candidates that are the same as the complex word. We trained the model on a machine with an NVidia RTX 3090, Intel core i9 CPU, with 32G of RAM. It took around 2 hours for the whole process: the training and the evaluation on the three datasets.

[6]https://www.pytorchlightning.ai