Reproducibility of *Exploring Neural Text Simplification Models*: A Review

Mohammad Arvan, Luís Pina, and **Natalie Parde**
Department of Computer Science
University of Illinois Chicago
{marvan3,luispina,parde}@uic.edu

Abstract

The reproducibility of NLP research has drawn increased attention over the last few years. Several tools, guidelines, and metrics have been introduced to address concerns in regard to this problem; however, much work still remains to ensure widespread adoption of effective reproducibility standards. In this work, we review the reproducibility of *Exploring Neural Text Simplification Models* by Nisioi et al. (2017), evaluating it from three main aspects: data, software artifacts, and automatic evaluations. We discuss the challenges and issues we faced during this process. Furthermore, we explore the adequacy of current reproducibility standards. Our code, trained models, and a docker container of the environment used for training and evaluation are made publicly available.

1 Introduction

In a survey conducted among 1,576 scientific researchers by Nature (Baker, 2016), 90% believed that there is at least a slight crisis when it comes to the reproducibility of research. Although there are no concrete statistics, the quantity and the growth of machine learning publications that rely on empirical evidence have recently raised alarms. To improve reproducibility, this community has designed checklists (AAAI, 2022; ACL, 2022; Deutsch et al., 2022), guidelines (ACM, 2022), and challenges (Sinha et al., 2022; Belz et al., 2021), which highlight the importance of reproducibility, encourage best practices, and create a platform for conducting reproducibility studies.

Still, measures of reproducibility "in the wild" (that is, pertaining to real, widely cited machine learning and natural language processing studies) are limited. In this work, we set out to reproduce one such study as a case example to provide a concrete measure of reproducibility for a specific work. We select *Exploring Neural Text Simplification Models* by Nisioi et al. (2017). This paper poses an intriguing case: the research artifacts released by the authors are of high quality, the details they have provided match or exceed the current reproducibility recommendations, and two other reproducibility studies (Cooper and Shardlow, 2020; Belz et al., 2022) have successfully reproduced the results with high precision. Reviewing a high-quality scientific publication that has been the focus of multiple reproducibility studies enables us to build and expand upon those works.

Our primary objective is to investigate the ease with and extent to which the selected paper can be reproduced. We limit introducing new configurations, adding them only to cases necessary for further understanding the reproducibility results and not for competing scenarios. In Section 2, we present the background, the task itself, the model, and its variants. In Section 3, we describe our methodology and the steps we take to review the reproducibility of Nisioi et al. (2017)'s work. In particular, we look at associated data, software artifacts, and automatic evaluations. We present our results in Section 4, before concluding by discussing our findings and recommendations for addressing the shortcomings of current checklists (Section 5). We release our reproducibility artifacts to facilitate and promote future reproducibility studies (Arvan et al., 2022). These artifacts include the updated source code, trained model, and complete runtime environment in a self-contained docker container.

2 Neural Text Simplification

Nisioi et al. (2017)'s work explores the task of *neural text simplification*. In this task, the goal is to transform a given text into a simpler version while retaining its meaning. What constitutes simplicity itself raises complicated questions since simplicity could be observed in the form of lexical simplification, content reduction, and grammatical or structural modification. Data-driven techniques attempt to achieve simplicity through automated metrics

Proceedings of the 15th International Conference on Natural Language Generation: Generation Challenges, pages 62 - 70

and human evaluation. The task holds many parallels with machine translation (MT), and this framing allows models studied in the context of neural MT (e.g., neural sequence to sequence models) to be adapted and deployed for neural text simplification.

Nisioi et al. (2017)'s work is one of the first investigations of neural sequence to sequence models for automatic text simplification. In particular, they use Long Short-Term Memory (LSTM) networks (Hochreiter and Schmidhuber, 1996, 1997) in an encoder-decoder architecture that has demonstrated success in similar sequence to sequence problems (Luong et al., 2015). The encoder LSTM computes a representation for each source sentence, and the decoder LSTM generates an output given the encoded representation and previously generated tokens. Nisioi et al. (2017) also employ a global attention mechanism that provides a more dynamic information flow and increases the representation bandwidth. To avoid overfitting, they use dropout (Srivastava et al., 2014), a technique that injects noise into the input during training by masking out certain features.

Nisioi et al. (2017) experiment with two variants of networks: one with random embedding weight initialization (*NTS*), and another with pre-trained embeddings. The latter is built by concatenating pre-trained word2vec embeddings from the Google News corpus (Mikolov et al., 2013a) with a locally trained skip-gram model (Mikolov et al., 2013b) with hierarchical softmax and a window size of 10. The concatenation process involves utilizing a unique dictionary associated with the source and target embeddings. The authors refer to this variant as *NTS w2v*.

3 Methodology

There is no standard protocol or set of guidelines for conducting a reproducibility study, so we rely on the best practices suggested by others to provide a subjective and objective evaluation of the reproducibility of Nisioi et al. (2017)'s work. These best practices originate from checklists (Pineau et al., 2021), research focused on reporting and evaluation (Dodge et al., 2019; Patterson et al., 2021; Schwartz et al., 2020), and reproducibility tracks at conferences and workshops (Deutsch et al., 2022). Although we do not fill out any checklists, as they are not created for the purpose of third-party evaluation, we cover nearly all the concerns they attempt to address.

Split	Sentences
train (EW-SEW)	284,677
validation (TurkCorpus)	2,000
test (TurkCorpus)	359

Table 1: Distribution of sentence pairs across different data splits, with data sources in parentheses.

In the following subsections, we first examine the data used for this work and then shift our attention to the released software artifacts. We perform these preliminary steps to identify potential obstacles to reproducibility and to test the adequacy of the existing standards. Later, we assess the reproducibility of the reported automatic evaluations.

3.1 Data

Data quality and composition are primary factors that can significantly impact reported results. Unsurprisingly, all reproducibility checklists emphasize the importance of data transparency. Nisioi et al. (2017) used a corpus of parallel English Wikipedia and Simple English Wikipedia (EW-SEW) articles (Hwang et al., 2015) when developing and evaluating their text simplification model. EW-SEW includes both manually and automatically aligned sentence pairs and was one of the largest publicly available datasets for text simplification at the time Nisioi et al. (2017)'s paper was published. Sentences in EW-SEW were filtered based on Wiktionary-based word-level semantic similarity scores included in the dataset, with a retention threshold set at 0.45. This resulted in a final set of 280K+ aligned sentences. EW-SEW does not have standard validation and test splits; thus, although Nisioi et al. (2017) used EW-SEW for training, they used TurkCorpus (Xu et al., 2016) for validation and testing. TurkCorpus is considerably smaller than EW-SEW and consists of 2000 validation and 359 test sentences.

The final distribution of training, validation, and test data is shown in Table 1. To preprocess the data, Nisioi et al. (2017) used the Stanford named entity recognition (NER) system (Finkel et al., 2005) to automatically tag the locations, persons, organizations, and miscellaneous entities in the dataset. We check the reproducibility of the preprocessing steps in Subsection 4.1 by reviewing the original dataset, as well as steps taken to filter and process the data.

3.2 Software Artifacts

Authors may omit purportedly trivial details from research publications due to strict length limits. Such details may be crucial for later successful replication. Fortunately, released software artifacts often provide these details and other necessary engineering steps. The ML Completeness Checklist (Stojnic, 2022) underlines the inclusion of five items in software artifacts that facilitate reproducibility and are expected to result in easier adaptability for future researchers: (1) specification of dependencies, (2) training code, (3) evaluation code, (4) pre-trained models, and (5) a README file including a table of results accompanied by precise commands to run and produce those results. Given that Nisioi et al. (2017) provided all of these items, we investigate the quality and the functionality of the released artifacts within this context by reviewing the aforementioned checklist items, testing out the provided commands, and rebuilding the environment using provided materials.

3.3 Automatic Evaluation

Reproducibility and reporting quality are complementary to one another, and improvements to one often accompany improvements to the other. We include task-agnostic metrics and details commonly used in training and evaluations of neural networks (Dodge et al., 2019; ACL, 2022) in our assessment of the reproducibility of Nisioi et al. (2017)'s automated evaluations. Namely, we check the number of parameters in the model, the computing infrastructure used to achieve results, and the total GPU hours required to train the model. We also report the model's total floating-point operations (fpo) (Schwartz et al., 2020), providing an estimate of the amount of computational work performed irrespective of the hardware setting. In neural networks, the dominant floating-point operations are ADD and MUL operations performed by a GPU.

Nisioi et al. (2017) evaluated the performance of their neural text simplification approach using two automated metrics as well as a human performance assessment. Their automated metrics included BLEU (Papineni et al., 2002; Wagner, 2010), a precision-based metric commonly used for machine translation and text simplification; and SARI (Xu et al., 2016), a metric designed specifically for text simplification that compares the system output against reference output and the input sentence. The evaluation scripts for these metrics are included in the source code released by the Nisioi et al. (2017). In addition to calculating the BLEU score using the script provided by Nisioi et al. (2017), we also calculate it using sacreBLEU v2.1,[1] a Python library that aims to unify standards for calculating the BLEU score (Post, 2018).

Ultimately, these metrics are used on various output files generated by different variants. At first, we evaluate the original outputs provided by Nisioi et al. (2017). Then, we use the trained model released by the authors to generate a new output and evaluate it using the mentioned metrics. Lastly, we use the code and the configuration provided by the authors in their publication and in their source code to train new models. Using these newly trained models, we generate yet another set of outputs. During this process, we are reducing the set of controlled conditions affecting the final results. We expected variation to increase as fewer conditions are controlled.

We used these metrics to evaluate the performance of our reproduced model, facilitating a direct comparison with the originally reported performance. Instead of viewing the reproducibility of the automatic evaluations as a binary state, e.g., reproducible or not reproducible, we use Quantified Reproducibility Assessment (QRA), a framework proposed by Belz et al. (2022). This framework defines reproducibility as a condition of measurement, out of a set of conditions that includes different locations, operators, and measuring systems, among other variables. As a result, identifying and reporting such conditions is an important part of this framework. Belz et al. (2022) quantify reproducibility as a measurement of precision. Given a different set of empirical results in this paper and previously conducted reproducibility studies, we present the coefficient of variation with small sample correction (CV^*) associated with each variant.

We take two additional steps to verify the claims based on the empirical results. At first, we use paired bootstrap resampling (Koehn, 2004) with 1000 samples to compare the performance of the two main variants on the output files released by Nisioi et al. (2017). Lastly, considering the relatively small size of the datasets used for training, validation, and testing, we suspected the random seed may greatly impact results. Therefore, we designed an experiment to quantify its impact. We

[1] https://github.com/mjpost/sacrebleu

trained 36 models [2] with the same configuration but different unique random seeds. A small variation in the final results of this experiment suggests that the effect of the random seed is negligible.

4 Results

In this section, we describe the outcomes of our reproducibility study in terms of data, software artifacts, and automatic evaluation.

4.1 Data

We were unable to analyze the original unfiltered version of the EW-SEW dataset (Hwang et al., 2015) as planned because the webpage containing the dataset no longer exists,[3] nor could earlier versions be retrieved using web archival tools (e.g., the Wayback Machine[4]). The released code repository for the selected paper also does not include scripts for filtering the dataset. As such, we could not review or reproduce the authors' preprocessing steps. However, the code repository does contain preprocessed dataset files, which allowed us to perform all other steps of our reproducibility analysis.

4.2 Software Artifacts

As mentioned earlier, the authors released a five star repository according to the ML Completeness Checklist. The authors listed the required external libraries, as well as Python- and Lua-specific dependencies. Moreover, the authors included a dockerfile containing the computing environment used for the experiments. Unfortunately, since a self-contained docker container was not included, it is not possible to rebuild the dockerfile, and most dependencies have been deprecated for years. These dependencies include Ubuntu 14.04 with an end of life (EOL) of 2019, Python 2.7 with EOL of 2020, Torch7 with last active development of 2017, and OpenNMT made obsolete in 2018 due to lack of support for Torch7, among others. Ultimately, we switched to another docker image based on Nvidia's CUDA 10.1 images that comes with Torch7 installed. This introduced further complications as recently released GPUs (e.g., those in the RTX 3000 series) require CUDA 11 or higher. We avoided this problem for now, but fixing this problem (which is beyond the scope of our present work) requires porting Torch7 and rebuilding it using the appropriate CUDA toolkit.[5]

Aside from the initial hurdle to get the repository to a running state, we did not face any major issues in using the software artifacts. Nisioi et al. (2017) provided the training code, evaluation code, and pre-trained models.[6] The README file contains instructions and required commands to produce the reported results. There were a few minor discrepancies between the provided instructions and real-world use, but we managed to resolve these issues. We note that the repository does not contain all configuration files used for each model variant. Hence, we use the information provided in the paper to recreate those.

In reviewing the source code, we found three issues affecting *NTS w2v* variants. We contacted the authors regarding these issues, and they graciously confirmed the first two. At the time of writing this paper, we still have not heard back regarding the third reported issue. We intend to investigate the impact of the first two issues on the results. These issues are described below.

4.2.1 Issue 1: Data Contamination

The *NTS w2v* models use a multi-step process to concatenate the pre-trained Google News word2vec embeddings and another embedding trained by the authors using the skip-gram technique. We found that during the skip-gram training process, this embedding utilized all datasets (including the development and test set), introducing data contamination that may call into question those models' results. The models affected by this issue are expected to have an advantage over other models. However, the validation and test sets are many times smaller than the training set, so performance gains may be negligible to non-existent.

4.2.2 Issue 2: Mismatched Embedding

This issue occurs during the concatenation process itself. This process uses two dictionaries, one for the encoder and one for the decoder, to generate the embedding matrix. However, we found that these embeddings were mismatched: the encoder used the decoder's dictionary, and the decoder used the encoder's dictionary. We expect fixing this issue will improve the performance of affected models.

[2] 14 training jobs failed after running out of storage.

[3] https://crow.ece.uw.edu/tial/projects/simplification/

[4] https://archive.org/web/

[5] Issue is reported here: https://github.com/nagadomi/distro/issues/11.

[6] https://github.com/senisioi/NeuralTextSimplification

System	BLEU ($\mu \pm 95\%$ CI)
Baseline: NTS w2v	87.9 (87.9 ± 2.0)
NTS	84.6 (84.6 ± 2.9)

Table 2: Statistical significance analysis performed on Nisioi et al. (2017)'s released output. With $p = 0.0079$, the difference in reported results between the two variants is statistically significant.

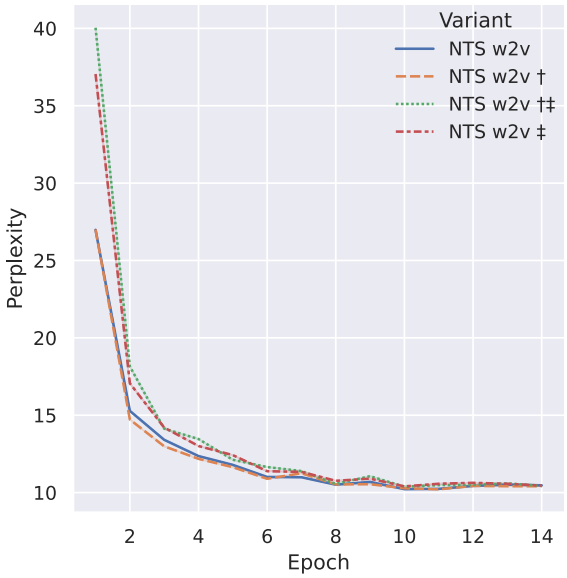

Figure 1: Validation perplexity of *NTS w2v* variants during training (lower is better). † indicates contaminated conditions, and ‡ indicates mismatched conditions.

4.2.3 Issue 3: Zero Embedding Weight

Lastly, we found that the final embedding matrix is missing the concatenation step, which results in zero vectors for all the words. Using a zero embedding weight nullifies the embedding pre-training altogether.

4.3 Automatic Evaluation

We follow the exact training setup provided by Nisioi et al. (2017), training models for 15 epochs with early stopping applied. Unlike the original paper, we did not tune the model using SARI or BLEU, and used the validation perplexity (lower is better) for model selection and early stopping. The translation is performed using beam search. Beam search generates the first k hypotheses at each step sorted by log-likelihood of the target sentence given the input sentence. While the authors experimented with using beam sizes of 5 and 12 and various hypotheses, we limit the scope of our experiments to 5 beams and 1 hypothesis. The hardware used for the experiments in the original paper is not explicitly specified. In our case, we use an *RTX 2080 ti* GPU to train the models. Training took approximately 3 hours. The model had 84 million parameters, of which 50 million belong to the embedding layer. With a maximum sequence length of 80 and a batch size of 1, this model used roughly 3G fpo (3×10^9) in a forward pass.

Table 3 contains the results of the original work (Nisioi et al., 2017) referred to as $t1$, reproducibility studies of Cooper and Shardlow (2020) ($t2$) and Belz et al. (2022) ($t3$), and the results calculated by this paper ($t4$) with their associated conditions. To ease the analysis, the results of every variant, measure, and output are grouped together. With the two evaluation scripts for calculating BLEU, we have added six values for BLEU and three for SARI. To be more specific, we have added automatic evaluation results for the output generated by Nisioi et al. (2017) or $o1$, our own output generated by running the trained model provided by Nisioi et al. (2017) or $o4$, and our own output generated by running our own version of the model or $o5$. We note that the model that we trained uses a source with all the fixes applied; however, to the best of our knowledge, all the other *NTS w2v* variants are trained with the mentioned issues. We present the precision results of the QRA framework in Table 4.

The *NTS* variant has CV^* values of 1.92 and 1.94 for SARI and BLEU, respectively. With 3.28 and 2.85 for SARI and BLEU, CV^* values for *NTS w2v* are slightly worse. However, the BLEU score of the *NTS w2v* variant reported by Cooper and Shardlow (2020) seems to be an outlier. By excluding their score (80.75), CV^* reduces to 1.22. There are two other interesting observations in Table 3. First, our reported results for $o1$ exactly match the reported results by the original paper; this suggests that we successfully recreated the environment they used for their evaluation. Second, the difference between the reported BLEU for $o1$ using the sacreBLEU evaluation script (Belz et al., 2022) and that found by our study implies there are still several unaccounted factors. We believe the version of sacreBLEU and the process of running this evaluation script are possible causes for this variation.

Table 2 shows results from the paired bootstrap resampling statistical significance test, with an objective of determining whether the performance of *NTS w2v* in terms of BLEU score is better than the

Object	Measurand	Output	Trained by	Comp. by	Eval. Script by	Performed by	Measured Value
NTS	BLEU	o1	t1	t1	t1	t1	84.51
		o1	t1	t1	t1	t2	84.50
		o1	t1	t1	≈t1	t3	85.60
		o1	t1	t1	sb	t3	84.20
		o1	t1	t1	t1	t4	84.51
		o1	t1	t1	sb2.1	t4	84.60
		o2	t2	t2	t1	t2	87.46
		o3	t1	t3	≈t1	t3	86.61
		o3	t1	t3	sb	t3	86.20
		o4	t1	t4	t1	t4	86.53
		o4	t1	t4	sb2.1	t4	86.60
		o5	t4	t4	t1	t4	88.81
		o5	t4	t4	sb2.1	t4	88.80
	SARI	o1	t1	t1	t1	t1	30.65
		o1	t1	t1	t1	t2	30.65
		o1	t1	t1	t1	t3	30.65
		o1	t1	t1	t1	t4	30.65
		o2	t2	t2	t1	t2	29.13
		o3	t1	t3	t1	t3	29.96
		o4	t1	t4	t1	t4	29.96
		o5	t4	t4	t1	t4	30.23
NTS w2v	BLEU	o1	t1	t1	t1	t1	87.50
		o1	t1	t1	≈t1	t3	89.36
		o1	t1	t1	sb	t3	88.10
		o1	t1	t1	t1	t4	87.50
		o1	t1	t1	sb2.1	t4	87.90
		o2	t2	t2	t1	t2	80.75
		o3	t1	t3	≈t1	t3	89.64
		o3	t1	t3	sb	t3	88.80
		o4	t1	t4	t1	t4	89.40
		o4	t1	t4	sb2.1	t4	89.40
		o5	t4	t4	t1	t4	87.04
		o5	t4	t4	sb2.1	t4	87.10
	SARI	o1	t1	t1	t1	t1	31.11
		o1	t1	t1	t1	t3	31.11
		o1	t1	t1	t1	t4	31.11
		o2	t2	t2	t1	t2	30.28
		o3	t1	t3	t1	t3	29.12
		o4	t1	t4	t1	t4	29.12
		o5	t4	t4	t1	t4	29.70

Table 3: Detailed overview of the results of *NTS* and *NTS-w2v*. All of the results utilize the source code released by Nisioi et al. (2017). Outputs $o1$ to $o5$ are generated based on the conditions provided in their respected row: $t1$=Nisioi et al. (2017), $t2$=Cooper and Shardlow (2020), $t3$=Belz et al. (2022), and $t4$= this paper; and sacreBLEU versions are represented as sb=unknown version, and $sb2.1$=version 2.1.

Object	Measurand	Sample Size	Mean	Unbiased STDEV	STDEV 95% CI	CV^*
NTS	SARI	8	30.23	0.56	[0.23, 0.89]	1.92
NTS	BLEU	13	86.07	1.64	[0.94, 2.34]	1.94
NTS w2v	SARI	7	30.22	0.96	[0.34, 1.58]	3.28
NTS w2v	BLEU	12	87.71	2.45	[1.35, 3.54]	2.85

Table 4: Precision (CV^*) and component measures (mean, standard deviation, standard deviation confidence intervals) for measured quantity values obtained in multiple measurements of the two *NTS* systems.

NTS variant. With $p = 0.0079$, the difference is indeed statistically significant. Since the output of *NTS w2v* is generated using a model affected by the zero weight embedding issue (Issue 3 described in Subsection 4.2.3), these two variants are essentially the same. Thus, understanding what is at play here requires assessing the results in Table 5. Even with a small sample size of 36, we observe values ranging from 84.47 to 89.59. This suggests that the performance difference between the two main

Measurand	Mean	Min	Max
SARI	29.24 ± 0.31	28.62	29.89
BLEU	87.9 ± 1.18	84.47	89.59

Table 5: Results of the random seed experiments on the TurkCorpus (Xu et al., 2016) test set, with a sample size of 36. Models are trained with the same configuration, but have unique random seeds. The evaluation script by Nisioi et al. (2017) was used.

variants may have originated from having different random seeds, even at statistically significant levels.

Finally, we investigate the issues reported for the *NTS-w2v* variant. We exclude Issue 4.2.3, as it simply converts *NTS-w2v* to *NTS* with zero embedding weight. We introduce three new variants:

- *NTS w2v*†: *NTS-w2v* only affected by data contamination (Issue 4.2.1).
- *NTS w2v*‡: *NTS-w2v* only affected by mismatched embeddings (Issue 4.2.2).
- *NTS w2v*†‡: *NTS-w2v* affected by data contamination and mismatched embeddings.

The results are shown in Table 6. Overall, the results are extremely close. We found that the variant with the data contaminated outperform others while *NTS-w2v*, the variant without any issues performed worse than the rest. We expected to observe a noticeable performance difference for the models affected by the mismatched embedding issue, but the performance gap was ultimately marginal and inconsistent. We report the validation performance during training to analyze whether there are any differences between these four variants. As shown in Figure 1, the models with mismatched embeddings had a worst start, by a perplexity gap of almost 15; however, as training progressed, they closed the gap and ended with perplexity differences of less than 1.

5 Discussion

Taking all our experiments into account, we cannot claim that the performance difference between different variants comes from the design decisions made during their development. The random seed and its cascading impact on weight initialization, data order, and sampling during text generation could be the primary cause of the observed variations. Similar to our work, Dodge et al. (2020)

Object	Measurand	Eval. Script by	Measured Value
NTS w2v	BLEU	t1	87.04
NTS w2v	BLEU	sb2.1	87.10
NTS w2v	SARI	t1	29.70
NTS w2v †	BLEU	t1	89.43
NTS w2v †	BLEU	sb2.1	89.40
NTS w2v †	SARI	t1	29.80
NTS w2v †‡	BLEU	t1	89.12
NTS w2v †‡	BLEU	sb2.1	89.10
NTS w2v †‡	SARI	t1	29.58
NTS w2v ‡	BLEU	t1	88.01
NTS w2v ‡	BLEU	sb2.1	88.00
NTS w2v ‡	SARI	t1	29.18

Table 6: Results of the experiments tracking performance impacts for identified issues, computed for this paper using our version of the model, our output, and the evaluation script provided by Nisioi et al. (2017) and sacreBLEU. † indicates contaminated conditions, and ‡ indicates mismatched conditions.

have observed that changes to random seeds can result in substantially different results.

Perhaps our most surprising finding is that the *NTS-w2v* variants affected by mismatched embeddings performed on par with the other variants once training was complete. This extreme level of resilience is, in fact, quite alarming. Nearly all publications utilizing neural networks report top-performing empirical results; yet, aside from manual code review and deep analysis of the final results, there are no other clear signs or warnings that may suggest a bug is impacting the model. In this case, we found that our findings from the random seed experiments and validation performance during the training process were the only indicators that something was amiss. We recommend that future studies include random seed analysis demonstrating the range of the results that can be achieved with varied seeds, although we recognize that this search is the most expensive in terms of computation and may not be feasible in every case.

Perhaps due to the age of this repository, getting the project to a running state consumed the most time. We suspect that the situation will deteriorate as most dependencies are no longer being actively maintained. Researchers should be hesitant with introducing new dependencies into their projects. Additionally, we believe it would be fruitful to redirect the time and effort used for identifying and reporting dependencies toward exporting self-contained environments. This is an inadequacy that we found in nearly all of the checklists; in the case of this

project, even though we knew all the requirements, we spent hours debugging different errors.

The reproducibility of a reproducibility study is equally important, if not more than the reported findings. While the contribution of the original paper includes a novel idea, our goal was to provide a final artifact having the highest possible degree of reproducibility, and to assess the ease with and extent to which the selected paper could be reproduced. It would be interesting to return and perform a meta-analysis of this work in a few years to see how much the claims hold over time. While it is impossible to stop hardware and software from changing constantly, there are steps that can be taken in order to prolong the lifespan of a research artifact. We have made changes and fixes publicly available in a forked repository of the original paper.[7] Additionally, we exported and released a self-contained docker container capable of training and running the model without any internet access (Arvan et al., 2022). Lastly, all the trained models are available for download.[8] Despite all these attempts, it is hard to predict future problems that might occur. Even the docker container depends upon the host environment (particularly, the kernel, GPU driver, and the docker itself). We have released our full runtime environment through Zendo (Arvan et al., 2022).

6 Conclusions

In this paper, we reviewed the reproducibility of *Exploring Neural Text Simplification Models* by Nisioi et al. (2017). In our three step process, we analyzed the reproducibility of the data, the software artifacts, and the automatic evaluations. We would have liked to analyze the reproducibility of human evaluations given additional time. We hope that our released artifacts offer other researchers a head start for future reproducibility studies.

7 Acknowledgments

We would like to thank Sergiu Nisioi, Maja Popovic, and Anya Belz for their excellent communication and collaboration. They provided further context regarding their experiments that cleared any lingering doubts we had regarding our results. This work would have not been the same without their help. We also thank the anonymous reviewers for their insightful feedback, which was incorporated into the final version of this paper. This work was partially supported by equipment purchased using the University of Illinois Chicago's Provost's Graduate Research Award.

[7] https://github.com/mo-arvan/NeuralTextSimplification

[8] https://drive.google.com/drive/folders/1cDQL08xQjuttu_jxrp1WnXKWZvEaRpSf

References

AAAI. 2022. AAAI reproducibility checklist.

ACL. 2022. ACL responsible nlp research.

ACM. 2022. ACM artifact review and badging.

Mohammad Arvan, Luís Pina, and Natalie Parde. 2022. Artifacts of Reproducibility of Exploring Neural Text Simplification Models: A Review.

Monya Baker. 2016. 1,500 scientists lift the lid on reproducibility. *Nature*, 533(7604).

Anya Belz, Maja Popovic, and Simon Mille. 2022. Quantified reproducibility assessment of NLP results. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 16–28, Dublin, Ireland. Association for Computational Linguistics.

Anya Belz, Anastasia Shimorina, Shubham Agarwal, and Ehud Reiter. 2021. The reprogen shared task on reproducibility of human evaluations in NLG: overview and results. In *Proceedings of the 14th International Conference on Natural Language Generation, INLG 2021, Aberdeen, Scotland, UK, 20-24 September, 2021*, pages 249–258. Association for Computational Linguistics.

Michael Cooper and Matthew Shardlow. 2020. Combinmt: An exploration into neural text simplification models. In *Proceedings of The 12th Language Resources and Evaluation Conference, LREC 2020, Marseille, France, May 11-16, 2020*, pages 5588–5594. European Language Resources Association.

Daniel Deutsch, Yash Kumar Lal, Annie Louis, Pete Walsh, Jesse Dodge, and Niranjan Balasubramanian. 2022. 2022 north american chapter of the association for computational linguistics reproducibility track.

Jesse Dodge, Suchin Gururangan, Dallas Card, Roy Schwartz, and Noah A. Smith. 2019. Show your work: Improved reporting of experimental results. 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 2019, Hong Kong, China, November 3-7, 2019*, pages 2185–2194. Association for Computational Linguistics.

Jesse Dodge, Gabriel Ilharco, Roy Schwartz, Ali Farhadi, Hannaneh Hajishirzi, and Noah A. Smith.

2020. Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping. *CoRR*, abs/2002.06305.

Jenny Rose Finkel, Trond Grenager, and Christopher D. Manning. 2005. Incorporating non-local information into information extraction systems by gibbs sampling. In *ACL 2005, 43rd Annual Meeting of the Association for Computational Linguistics, Proceedings of the Conference, 25-30 June 2005, University of Michigan, USA*, pages 363–370. The Association for Computer Linguistics.

Sepp Hochreiter and Jürgen Schmidhuber. 1996. LSTM can solve hard long time lag problems. In *Advances in Neural Information Processing Systems 9, NIPS, Denver, CO, USA, December 2-5, 1996*, pages 473–479. MIT Press.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

William Hwang, Hannaneh Hajishirzi, Mari Ostendorf, and Wei Wu. 2015. Aligning sentences from standard wikipedia to simple wikipedia. In *NAACL HLT 2015, The 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Denver, Colorado, USA, May 31 - June 5, 2015*, pages 211–217. The Association for Computational Linguistics.

Philipp Koehn. 2004. Statistical significance tests for machine translation evaluation. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004, 25-26 July 2004, Barcelona, Spain*, pages 388–395. ACL.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP 2015, Lisbon, Portugal, September 17-21, 2015*, pages 1412–1421. The Association for Computational Linguistics.

Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013a. Efficient estimation of word representations in vector space. In *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*.

Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. 2013b. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119.

Sergiu Nisioi, Sanja Stajner, Simone Paolo Ponzetto, and Liviu P. Dinu. 2017. Exploring neural text simplification models. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 2: Short Papers*, pages 85–91. Association for Computational Linguistics.

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, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

David A. Patterson, Joseph Gonzalez, Quoc V. Le, Chen Liang, Lluis-Miquel Munguia, Daniel Rothchild, David R. So, Maud Texier, and Jeff Dean. 2021. Carbon emissions and large neural network training. *CoRR*, abs/2104.10350.

Joelle Pineau, Philippe Vincent-Lamarre, Koustuv Sinha, Vincent Larivière, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Hugo Larochelle. 2021. Improving reproducibility in machine learning research(a report from the neurips 2019 reproducibility program). *J. Mach. Learn. Res.*, 22:164:1–164:20.

Matt Post. 2018. A call for clarity in reporting BLEU scores. In *Proceedings of the Third Conference on Machine Translation: Research Papers, WMT 2018, Belgium, Brussels, October 31 - November 1, 2018*, pages 186–191. Association for Computational Linguistics.

Roy Schwartz, Jesse Dodge, Noah A. Smith, and Oren Etzioni. 2020. Green AI. *Commun. ACM*, 63(12):54–63.

Koustuv Sinha, Jesse Dodge, Sasha Luccioni, Jessica Forde, Sharath Chandra Raparthy, François Mercier, Joelle Pineau, and Robert Stojnic. 2022. ML reproducibility challenge 2021.

Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. 2014. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1):1929–1958.

Robert Stojnic. 2022. ML code completeness checklist.

Wiebke Wagner. 2010. Steven bird, ewan klein and edward loper: Natural language processing with python, analyzing text with the natural language toolkit - o'reilly media, beijing, 2009, ISBN 978-0-596-51649-9. *Lang. Resour. Evaluation*, 44(4):421–424.

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