Reproducibility of Faithful Low-Resource Data-to-Text Generation through Cycle Training

Shang-Ling (Kate) Hsu[*] **Shicheng Wen**[*] **Shauryasikt Jena**[*]
Thomas Lord Department of Computer Science
University of Southern California
{hsushang, wenshich, jenas}@usc.edu

1 Introduction

In today's digital era, as data collection becomes more streamlined, databases are teeming with rich and intricate information. The evolving trend is not just to query this information directly, but to verbalize it in a manner suitable for interaction through conversational agents. While recent works in data-to-text generation have produced a variety of datasets, certain domains remain lacking in data, making it challenging to achieve high performance during fine-tuning.

To overcome the limitations of datasets and ensure consistency between data and text, Wang et al. (2023) present a novel contribution that introduces the application of cycle training (Iovine et al., 2022) to both data-to-text and text-to-data modeling using only a small amount of data. It leverages pre-trained T5 models (Raffel et al., 2020), which even rival the performance of supervised models in certain domains. This process ensures the faithfulness of the output text with respect to the input data and vice versa.

Furthermore, Wang et al. (2023) delves into an exhaustive empirical analysis, shedding light on the conditions favoring cycle training and scrutinizing the fidelity of data-to-text vis-à-vis various generative errors. Additionally, they introduce a novel counting and ranking annotation scheme, aiming to holistically evaluate the faithfulness of the generated text from the standpoints of correctness, faithfulness, data coverage, and fluency.

2 Scope of Reproducibility

Wang et al. (2023) propose the use of cycle training to enhance the faithfulness of text generation from structured and information-rich data. In our replication study, we focus on their assertion that cycle training, when initialized with a relatively small amount of supervised data (approximately 100 instances), achieves nearly identical performance to fully supervised methods in data-to-text generation. The core model used for this comparative analysis is the T5 text-to-text transfer transformer (Raffel et al., 2020).

We are motivated to reproduce these results because they highlight the performance of cycle training in generating high-quality textual output. This aligns with the central contribution of the original paper, which aims to generate coherent and correct text from information-rich data. We further apply cycle training to both data-to-text and text-to-data models, without resorting to graph-based methods, relying solely on the T5 model.

2.1 Tested Hypotheses from the Original Paper

Summarized below is a concise list of claims from the original paper (Wang et al., 2023) that we investigate:

- Unsupervised cycle training, when initialized with a limited amount (approximately 100 instances) of supervised data, replicates the performance of fully supervised methods in data-to-text generation. This is the central claim of the paper.

- Cycle training can be effectively applied to data-to-text models using a pre-trained T5 model without the need for graph-based methods or auxiliary models.

3 Methodology

3.1 Model description

Backbone Model The backbone model of the work used for both RDF-to-text and text-to-RDF is T5-base (Raffel et al., 2020) with 12 layers, a hidden size of 768, 12 self-attention heads, and 220M trainable parameters. The RDF triples for each sample are transformed into a sequence, denoted as d. This sequence uses the `[S]`, `[P]`, and

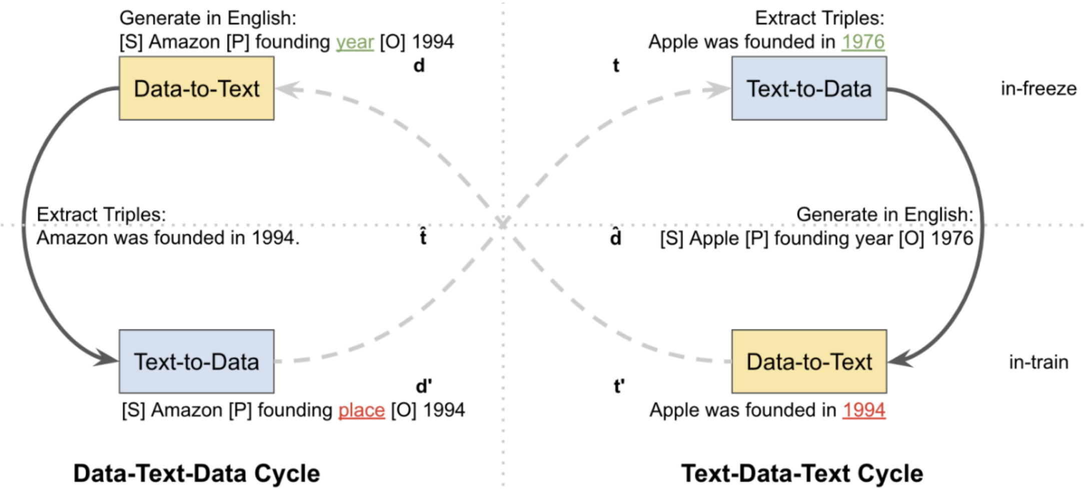
Generate in English:
[S] Amazon [P] founding year [O] 1994
d
Data-to-Text
Extract Triples:
Amazon was founded in 1994.
t̂
Text-to-Data
d'
[S] Amazon [P] founding place [O] 1994
Data-Text-Data Cycle
Extract Triples:
Apple was founded in 1976
t
Text-to-Data
in-freeze
Generate in English:
d̂
[S] Apple [P] founding year [O] 1976
Data-to-Text
in-train
t'
Apple was founded in 1994
Text-Data-Text Cycle

Figure 1: Schematic of cycle training (Wang et al., 2023)

[O] tags to represent the subject, predicate, and object of each triple, respectively. As a result, both RDF-to-text and text-to-RDF conversions can be approached and refined as sequence-to-sequence generation tasks.

BART-base (Lewis et al., 2019) with 12 layers, a hidden size of 768, 12 self-attention heads, and 139M trainable parameters was used as well, with the same preprocessing, to investigate the strength of the hypothesis on a different backbone.

BART was chosen to explore the generalizability of the method for a different backbone. Further, given its similar architecture to T5-base, the training hyperparameters need not be adjusted drastically either.

Cycle Training (Fig 1) The cycle training process comprises two cycles: Data-Text-Data (**DTD**) and Text-Data-Text (**TDT**). The **DTD** cycle emphasizes the self-consistency of data, while the **TDT** cycle similarly underscores the self-consistency of text. Within the **DTD** cycle, the Data-to-Text model accepts the linearized triples, denoted as d, producing the corresponding intermediate text, represented as $\hat{t}$. Subsequently, the Text-to-Data model aims to reconstitute d using the provided $\hat{t}$. The reconstruction loss, labeled as $L_{d'}$, is derived from the averaged negative log-likelihood:

$$\mathcal{L}_{d'} = -\frac{1}{|d|} \sum\nolimits_{i=0}^{|d|} \log p(d_i|d_0, ..., d_{i-1}, \hat{t})$$

where d_i represents the i-th token of sequence t, and $|d|$ designates the length of the sequence. Conversely, within the **TDT** cycle, the Text-to-Data model initiates by accepting the text, represented as t, and subsequently produces the corresponding linearized triples, denoted as $\hat{d}$. Following this, the Data-to-Text model is trained aiming to reconstruct t using the provided $\hat{d}$. The reconstruction loss, termed $L_{t'}$, is determined by the averaged negative log-likelihood. As illustrated below, t_i signifies the i-th token of sequence t, while $|t|$ represents the length of that sequence:

$$\mathcal{L}_{t'} = -\frac{1}{|d|} \sum\nolimits_{i=0}^{|t|} \log p(t_i|t_0, ..., t_{i-1}, \hat{d})$$

In each epoch, while one model is training, the other model will be frozen.

3.2 Data description

We obtained both the data and train/dev/test splits of the two datasets the paper used, which are Version 3.0, English version of WebNLG[1] (Gardent et al., 2017) and DART[2] (Radev et al., 2020), from HuggingFace. The authors divide DART data into three subsets, E2E, WTQ, and WSQL, according to the source attribute of the data, which is also available along with the dataset.

Beyond the original paper, for further exploration of how cycle training works on different domains of data, we utilized XAlign[3] (Abhishek et al., 2022) for comparison. The statistics of the datasets are shown in Table 1.

[1] https://huggingface.co/datasets/web_nlg

[2] https://huggingface.co/datasets/GEM/dart

[3] https://huggingface.co/datasets/tushar117/xalign/viewer/en

Dataset	Domain	Split Size (Lex) Train/Dev/Test	Unique Predicates	Triples/Sample median/max	Vocab Size	Tokens/Sample median/max
WebNLG	DBPedia (16 categories)	35,426/4,464/7,305	412	3 / 7	14,408	19 / 80
E2E	Restaurants	33,482/1,475/1,844	7	4 / 7	4,370	20 / 71
WTQ	Wikipedia (open-domain)	1,756/38/30	1,315	2 / 7	9,277	13 / 96
WSQL	Wikipedia (open-domain)	3,531/246/346	1,987	2 / 10	12,045	10 / 38
XAlign	Person biographies	5,000/500/470	213	2 / 20	21,063	16 / 70

Table 1: Datasets statistics and comparison

We used only the English split of XAlign and sampled 5,000 entries for the training set, 500 entries for the validation set, and 470 entries for the test set. Being from a different domain, XAlign is a useful dataset to be studied here, as it has the required S-P-O consistency. This dataset is instrumental in understanding the method's robustness being consistent for consistent data patterns.

3.3 Hyperparameters

We used similar hyperparameters as instructed in the paper — AdamW as optimizer with linear weight decay and 0 warmup steps, a max input length of 256, and a learning rate of 3×10^{-4}. Unlike the paper, we used a batch size of 32 instead of 256. We trained each model up to 50 epochs and patience of 5 epochs for early stopping. We selected the best model by METEOR score, and we did not use delta for this reproduction study. At inference time, we decode with the beam search algorithm using 4 beams and a generation length varying between 3 tokens and 256 tokens. We repeated the experiment of T5 as the backbone model 5 times for WebNLG, E2E, WTQ, and WSQL datasets using different random seeds, as well as for the XAlign dataset. We ran BART on one seed for all datasets as only an esimated pattern is expected to be observed.

100 annotated samples were chosen to initialize the cycle training to compare against the fully supervised method as suggested.

3.4 Implementation

The training and evaluation scripts, written in python, of this paper are publicly available.[4] All the non built-in modules it evokes are also publicly available: PyTorch, HuggingFace Transformers, Natural Language Toolkit (NLTK), HuggingFace Datasets, HuggingFace metrics, NumPy, and tqdm. Note that only the PARENT metric is not included in Hugging Face metrics or elsewhere. Hence, we will write our own data pre-processing, full finetuning, and the PARENT evaluation code and reuse only the code for cycle training and evaluation.

Dataset	Fully Unsupervised	Low-Resource Finetuning
WebNLG	36:43	0:57
E2E	24:58	0:43
WTQ	2:08	0:05
WSQL	3:30	0:07
XAlign	7:45	0:12

Table 2: Average computational time (hours:minutes) for training T5 on each dataset on A40 GPU

3.5 Experimental setup

We did the experiments in three different settings: full fine-tuning, unsupervised cycle training, and low-resource cycle training.

We ran our experiments on CARC[5] nodes with 8 CPUS and an NVIDIA A40 GPU. Our final code implementation is available at GitHub[6]. In total, we requested 78 nodes with different tasks.

3.6 Computational requirements

We requested cluster nodes for one day for each task of WTQ, WSQL, and XAlign datasets, and three days for WebNLG and E2E datasets. The cumulative computational resource request amounted to 136 days, utilizing 78 A40 nodes in total. It is noteworthy that the actual duration of training and evaluation processes was shorter than initially anticipated, due to the early stopping criteria. Table 2 shows the actual average computational time for each dataset in different settings over different random seeds.

[4] https://github.com/amzn/faithful-data2text-cycle-training

[5] https://www.carc.usc.edu/

[6] https://github.com/ktxlh/faithful-data2text-cycle-training

4 Results

To compare the performance of cycle training when initialized with a small number of annotated samples, we recorded the results on T5-base in Table 3, in accordance with (Wang et al., 2023).

The performance of the T5 model demonstrated consistency with the results of the original paper, even surpassing the numbers in certain metrics as reported in the original publication. The reproduction results also support the main idea of the original publication across datasets and metrics: Low-resource cycle training can achieve similar performance to fully-supervised fine-tuning. The T5 model's robust capability in summarization tasks translates into exceptional performance in the text-to-data aspect, making it more adept at learning text-to-data patterns. Leveraging more accurate intermediate outputs, cycle training can achieve superior performance on this foundation. It is noteworthy that on larger datasets (E2E, WebNLG), the low-resource fine-tuning with additional cycle training exhibited better performance compared to fully unsupervised cycle training methods, highlighting the positive impact of low-resource fine-tuning.

Beyond the original paper, Table 4 records the performances of different models trained on the BART-base backbone. The results for BART are not in as much agreement as are the results for T5. Notably, besides the WSQL dataset, the performance of the BART model significantly declines when additional cycle training is incorporated after low-resource fine-tuning.

5 Discussion

The results on the T5 backbone are mostly in agreement with the paper, thus implying the two key points of the hypothesis — unsupervised cycle training initialized with a few annotated samples can compete with the performance of fully supervised methods and that no graph-based methods or auxillary models were required in this unsupervised setup. This is extremely useful in domains where there is a severe lack of labeled data, but some labeled data is present with the S-P-O consistency. The small labeled portion can be used to initialize the cycle training as mentioned, and utilize unsupervised training over available data - a bottleneck of supervised learning.

However, the performances on the BART backbone remain uncomparable to the performances on T5 backbone. However, this cannot be strong evidence that BART cannot perform well on the task. The phenomenon of the negative impact of cycle training on low-resource fine-tuning might be because the cycle training disrupted the language expression capabilities that the model had learned during low-resource fine-tuning. Indicating that BART did not successfully learn data-to-text or text-to-data patterns in the initial cycle of training. For future exploration, we plan to adjust hyperparameters, such as reducing the learning rate. Such adjustments could help us to investigate more deeply the effects of additional cycle training in low-resource environments. Through this approach, we hope to better understand the potential and limitations of the BART model and achieve superior performance in data-to-text tasks.

Preprocessing the datasets was fairly uncomplicated, however lack of clarity regarding the statistical terms and the versions of the datasets in the original paper led to some discrepancies in values. Also, completing our code for fully supervised learning and fine-tuning was achievable, however much time could have been saved had these trivial scripts been readily available as well.

Reproducibility of the hypothesis could be improved if the cycle training code was made robust to accepting the backbone via inline arguments. Further, a study on the choice of hyperparameters given the model's architecture and problem (dataset) size would be beneficial practically and/or industrially. Finally, the usage of "low-resource cycle training" is misleading as one would assume that cycle training is utilizing a small amount of data. However, that is not the case as cycle training happens in an unsupervised manner over the whole dataset after the backbone has been finetuned over a small set of labeled samples.

In summary, a set of recommendations for improving reproducibility includes:

1. Providing the rigorous definitions of dataset statistics.
2. Clarifying the precise inclusion and exclusion criteria for data instances.
3. Open-sourcing an essential, end-to-end set of scripts that produce the key results of the paper.
4. Modularizing source code and enable interesting options to be arguments.

Method	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	BLEU	BertScore	PARENT
Tested on WebNLG							
Fully-supervised fine-tuning	71.68	46.27	**55.88**	52.26	32.36	94.02	51.29
Unsupervised cycle training	70.99(0.45)	45.09(0.41)	54.09(0.25)	51.65(0.35)	31.06(0.31)	93.78(0.07)	49.74(0.71)
Low-resource fine-tuning	63.90(0.48)	41.61(0.59)	53.81(0.58)	44.10(1.01)	23.72(1.08)	93.31(0.09)	
+ cycle training	**72.46**(0.24)	**46.65**(0.31)	55.77(0.19)	**52.95**(0.24)	**32.61**(0.22)	**94.04**(0.05)	**51.94**(0.65)
Tested on E2E							
Fully-supervised fine-tuning	**69.53**	**42.73**	50.87	53.39	29.46	94.25	54.05
Unsupervised cycle training	62.67(2.2)	36.67(2.26)	44.3(2.63)	49.88(2.87)	26.23(3.06)	93.06(0.34)	49.82(3.37)
Low-resource fine-tuning	66.99(0.36)	42.44(0.50)	**52.65**(0.37)	48.01(0.81)	27.21(0.84)	**94.29**(0.03)	
+ cycle training	69.47(0.34)	42.66(0.45)	50.70(0.59)	**53.41**(0.38)	**30.32**(0.59)	94.16(0.09)	**57.58**(0.38)
Tested on WTQ							
Fully-supervised fine-tuning	57.11	32.51	44.21	36.55	18.59	90.67	28.40
Unsupervised cycle training	**63.30**(0.99)	**40.04**(0.80)	**52.55**(0.67)	**45.18**(0.55)	**21.11**(1.72)	90.86(0.22)	**32.01**(1.56)
Low-resource fine-tuning	55.25(1.92)	31.33(1.53)	48.33(1.99)	36.75(2.03)	17.72(2.95)	**91.24**(0.09)	
+ cycle training	59.96(0.72)	31.54(1.06)	45.82(0.31)	37.06(0.40)	20.21(0.92)	90.65(0.10)	30.47(1.65)
Tested on WSQL							
Fully-supervised fine-tuning	63.02	**40.30**	53.18	43.44	21.50	91.02	27.15
Unsupervised cycle training	58.12(1.34)	30.41(0.97)	43.92(1.18)	35.58(0.66)	20.11(0.91)	90.48(0.31)	**31.25**(0.29)
Low-resource fine-tuning	61.04(1.12)	37.15(1.18)	**53.90**(1.23)	42.77(0.92)	20.45(0.63)	**91.22**(0.64)	
+ cycle training	**63.16**(0.39)	39.77(0.93)	52.44(0.47)	**45.12**(0.53)	**21.72**(3.03)	90.80(0.19)	31.24(0.63)
Tested on XAlign *							
Fully-supervised fine-tuning	**68.52**	**50.10**	**63.80**	**58.84**	44.19	**95.01**	54.01
Unsupervised cycle training	68.20	50.06	62.26	55.22	**46.95**	94.35	**60.25**
Low-resource fine-tuning	65.04(2.87)	44.92(4.49)	61.09(3.45)	50.50(2.88)	37.63(4.60)	93.92(0.89)	
+ cycle training	66.41(0.88)	48.72(0.63)	60.52(0.79)	54.98(0.36)	44.62(1.14)	94.17(0.14)	59.83(0.53)

Table 3: Data-to-Text generation performances on datasets over T5-base backbone, **bold:** best model over metric; numbers in parentheses denote the standard deviations across 5 runs; * denotes our additional dataset to the original publication.

5. Providing details about how the hyperparameters were tuned.

6. Fostering a more genuine and informative foundation in writing for readers to assess the feasibility of replication.

7. Sharing models and metrics via some popular platform, such as HuggingFace, to mitigate the barriers of reproduction.

6 Communication with original authors

We communicated with the original authors via emails and GitHub issues. We emailed the authors for the details in data processing because it is unclear in the paper. After 5 days, we received the first response, in which the authors suggested that we refer to the paper or open an issue on GitHub. In response, we opened a GitHub issue about dataset statistics reproduction[7] and replied to another GitHub issue about data processing details.[8][9] Although the authors are not able to provide more scripts or data samples due to their company's confidential policy, the first author of the paper provided prompt and informative responses to GitHub issues, eliminating most of our confusion.

[7] https://github.com/amzn/faithful-data2text-cycle-training/issues/2

[8] https://github.com/Edillower/CycleNLG/issues/1#issuecomment-1828800748

[9] The two issues are created in two different repositories, and both of which are source code of the paper.

Method	ROUGE-1	ROUGE-2	ROUGE-L	METEOR	BLEU	BertScore	PARENT
Tested on WebNLG							
Unsupervised cycle training	51.43	29.90	41.16	37.43	21.51	90.14	33.73
Low-resource fine-tuning	48.50	27.58	42.02	32.15	13.13	90.85	
+ cycle training	34.79	17.33	28.83	23.66	12.10	86.83	17.58
Tested on E2E							
Unsupervised cycle training	44.29	21.69	33.92	28.58	11.83	90.27	26.37
Low-resource fine-tuning	35.21	14.48	30.36	22.96	8.05	90.12	
+ cycle training	34.51	11.63	25.64	25.65	7.58	88.79	7.24
Tested on WTQ							
Unsupervised cycle training	47.05	23.99	37.36	27.87	16.07	89.22	26.15
Low-resource fine-tuning	23.14	8.39	21.15	14.78	5.06	86.52	
+ cycle training	12.11	5.29	9.99	8.57	0.00	83.11	0.15
Tested on WSQL							
Unsupervised cycle training	47.05	23.99	37.36	35.59	15.46	88.96	23.59
Low-resource fine-tuning	40.65	18.76	36.58	27.76	13.95	87.99	
+ cycle training	48.50	27.81	41.76	33.20	14.32	88.76	23.33

Table 4: Data-to-Text generation performances on datasets over BART-base backbone, **bold:** best model over metric

References

Tushar Abhishek, Shivprasad Sagare, Bhavyajeet Singh, Anubhav Sharma, Manish Gupta, and Vasudeva Varma. 2022. Xalign: Cross-lingual fact-to-text alignment and generation for low-resource languages. *arXiv preprint arXiv:2202.00291*.

Claire Gardent, Anastasia Shimorina, Shashi Narayan, and Laura Perez-Beltrachini. 2017. Creating training corpora for NLG micro-planners. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 179–188. Association for Computational Linguistics.

Andrea Iovine, Anjie Fang, Besnik Fetahu, Jie Zhao, Oleg Rokhlenko, and Shervin Malmasi. 2022. CycleKQR: Unsupervised bidirectional keyword-question rewriting. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11875–11886, Abu Dhabi, United Arab Emirates. 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.

Dragomir Radev, Rui Zhang, Amrit Rau, Abhinand Sivaprasad, Chiachun Hsieh, Nazneen Fatema Rajani, Xiangru Tang, Aadit Vyas, Neha Verma, Pranav Krishna, Yangxiaokang Liu, Nadia Irwanto, Jessica Pan, Faiaz Rahman, Ahmad Zaidi, Murori Mutuma, Yasin Tarabar, Ankit Gupta, Tao Yu, Yi Chern Tan, Xi Victoria Lin, Caiming Xiong, and Richard Socher. 2020. Dart: Open-domain structured data record to text generation. *arXiv preprint arXiv:2007.02871*.

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. *J. Mach. Learn. Res.*, 21(1).

Zhuoer Wang, Marcus Collins, Nikhita Vedula, Simone Filice, Shervin Malmasi, and Oleg Rokhlenko. 2023. Faithful low-resource data-to-text generation through cycle training. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2847–2867, Toronto, Canada. Association for Computational Linguistics.