Reproducibility Project: Detoxifying Text with MaRCo Controllable Revision with Experts and Anti-Experts

Daniel Pereira da Costa and **Ian Wu** and **David Chu**
University of Southern California
{danielp3, ianwu, mhchu}@usc.edu

1 Introduction

Handling toxic, offensive, and biased language poses significant challenges, most notably when it lacks explicit toxic keywords, impacting both online and offline communities. *Text detoxification*, the process of rewriting text to reduce toxicity while preserving its original meaning, offers a promising solution to combat this issue. Hallinan et al. (2023) proposes a novel algorithm for text detoxification called MaRCo. The MaRCo algorithm is an weakly-supervised text detoxification approach consisting of two key steps: masking, and context-aware token replacement throughout the sequence.

In the masking phase, MaRCo approach utilizes an unsupervised controlled revision based on autoencoder Language Models (AE-LM). Their setup includes a base pretrained AE-LM, an expert AE-LM fine-tuned on data with desirable attributes, and an anti-expert AE-LM fine-tuned on data with undesirable attributes. To locate potential toxic content in a text, they compare the probability distributions of each word under expert and anti-expert models. Tokens with significant differences in the Jensen-Shannon divergence distribution are masked as possibly toxic.

After identifying potentially toxic areas in the text, MaRCo replaces these sections with safer tokens. This approach adapts the DEXPERTS framework (Liu et al., 2021), which steers models away from generating toxic content through probability ensembling, to enable rewriting using AE-LMs. They obtain probability distributions for the next token from the base and expert AE-LMs. The base model replicates the input sequence but is guided by the expert and anti-expert at masked areas with contrasting probability distributions. This method allows for meaningful revisions while preserving non-toxic content.

2 Scope of reproducibility

2.1 Claims from the original paper

We test MaCRo's proclaimed ability to capture and rewrite subtle toxic text compared to previous methods that rely on toxicity classifiers or toxic word lists by attempting to verify the claims from Hallinan et al. (2023) that MaRCo outperforms state-of-the-art baselines on automatic metrics. Specifically, we want to reproduce automatic evaluations on detoxified generations on MAgr (Breitfeller et al., 2019), SBF (Sap et al., 2020), and DynaHate (Vidgen et al., 2021) datasets for MaRCo, ParaGeDi and CondBERT (more details in section 3).

The original work also utilized human evaluations through Amazon Mechanical Turk to compare the toxicity of MaRCo rewrites with those generated from prior methods. While this showed that evaluators generally preferred MaRCo rewrites, we do not reproduce this aspect of the work, as it is not a focal point of the evaluations and would be costly and highly time-intensive to reproduce.

In summary, we address the following claims:

1. The MaRCo algorithm consistently outperforms state-of-the-art detoxification baselines from previous work by Dale et al. (2021) in toxicity reduction.

2. MaRCo maintains the highest degree of fluency in its rewrites when compared to baseline approaches.

3. Semantic meaning is sufficiently preserved in MaRCo rewrites, as demonstrated by a high BERTScore against the original reference text.

2.2 Claims beyond the scope of the original paper

The authors originally evaluated the MaRCo algorithm using BART-Base, a 139M parameter mid-sized model, leading to high training times and computational requirements. Thus, we evaluate the need for a language model of this size. As the expert model is trained on a relatively large dataset, evaluating the efficacy of a smaller model has the potential to greatly decrease the computational requirements for implementing MaRCo. While the dataset for training the anti-expert model is significantly smaller (approximately 140k samples), we hypothesize that this may cause overfitting when used to train the BART-Base model and that comparable or even better results can be achieved with the use of a smaller model. To this end, we perform a 2 X 2 study, in which we evaluate the effects of using different combinations of model sizes (base + base, base + small, small + base, and small + small for expert and anti-expert models, respectively) on evaluation metrics. For the small model, we use an unofficial implementation of BART[1] from Hugging Face with approximately 70 million parameters, about half that of BART-base.

In addition, as large language models (LLMs) often achieve performance comparable to state-of-the-art methods in recent literature, we also evaluate their performance compared to MaRCo. Given their impressive capabilities in natural language understanding and generation (NLU/NLG), we expect LLMs to perform well in terms of fluency and semantic preservation in their rewrites, in addition to toxicity reduction. While we tested several LLMs (Llama-2, ChatGPT, GPT-4) for this task, they tended to be overly "censored" and would not complete the requested rewrites. Because of this, we use the Mistral-7B model through the Together.ai API, as it is known to be less "censored" when compared with other LLMs.

Finally, considering that the Perspective API, which was used for toxicity measurement in the original paper, often updates its models to newer versions, we suggest employing a transformer-based model known as Detoxify for assessing the detoxifying performance of the models. This approach aims to mitigate any discrepancies due to version updates and further strengthen the robustness of our toxicity evaluation.

3 Methodology

3.1 Model descriptions

To mask and replace toxic language, the MaRCo algorithm leverages an expert model and an anti-expert model. In the original work, these models are fine-tuned from the pre-trained BART-Base model (Lewis et al., 2020), with non-toxic and toxic data, respectively. Training is conducted to optimize for next-token prediction (conditional generation) with cross-entropy loss through supervised learning.

JigSaw	# rows	
	Train	Val
Toxic	115,216	29,118
Non-Toxic	1,009,610	225,154

Table 1: Summary of Training Datasets

BART-Base is a mid-sized model with approximately 139M parameters and achieves similar or better performance when compared to RoBERTa (Liu et al., 2019), a model with 355M parameters, on a variety of tasks.

3.1.1 Baseline Models

The following state-of-the-art detoxifying models from Dale et al. (2021) are used as baselines.

ParaGeDi incorporates a class-conditioned language model that operates on top of a paraphrasing language model to guide generated text toward a specific attribute.

CondBERT follows a pointwise editing approach. It begins by identifying tokens to mask and subsequently utilizes a mask-filling model to replace them. Unlike MaRCo, CondBERT employs a lexicon-based technique for masking words, utilizing weights derived from a whole-word toxic language logistic classifier.

3.2 Data descriptions

Both models are fine-tuned using over 1.8M human annotated comments from the Jigsaw Unintended Bias in Toxicity Classification Kaggle challenge[2], which consists of forum comments related to news articles. Each comment is annotated by a worker, who assigns a toxicity label (1 if the comment is considered toxic, 0 otherwise). Additionally, the comments are categorized into five toxicity sub-types: identity_attack, insult, obscene, sexual_explicit, and threat, with each sub-type, assigned a label of 1 if the worker identified the corresponding trait in the comment or 0 if not.

The Expert model was trained on 1 million non-toxic comments, while the Anti-Expert model utilized 115K toxic comments. The original paper did not specify the exact partitions used for training

[1]https://huggingface.co/lucadiliello/bart-small

[2]jigsaw-unintended-bias-in-toxicity-classification

Hyperparameter	Tested	Assignment
repetition penalty	[1.0, 1.2, 1.5]	1.0
α_1	[0, 0.5, 1.0, 1.5]	1.5
α_2	[3.0, 3.25,..., 5.0]	4.25
temperature (base model)	[0.9, 1.3, ..., 2.9]	2.5

Table 2: Hyperparameters tested and used for MaRCo on MAgr

Hyperparameter	Tested	Assignment
repetition penalty	[1.0, 1.2, 1.5]	1.5
α_1	[0, 0.5, 1.0, 1.5]	1.5
α_2	[3.0, 3.25,..., 5.0]	5.0
temperature (base model)	[0.9, 1.3, ..., 2.9]	2.9

Table 3: Hyperparameters tested and used for MaRCo on SBF

and evaluation of the Expert model, as they mentioned that providing them was not feasible due to the size of the dataset.

To replicate the training data, we followed the steps outlined in the paper. Comments with over 50% toxic annotations (where the 'toxic' column is equal to 1) were categorized as toxic, while comments without toxic annotations were classified as non-toxic. Since the authors only provided the toxic datasets, we verified our dataset against that portion, following the same train-validation split of 80% and 20%, respectively. However, since no information was available regarding the non-toxic dataset split, we also employed an 80/20 division for consistency. The details of the training and validation splits can be found in Table 1:

For evaluating the model performance, three out-of-domain datasets with subtle toxicity were used. The authors provided all 3 datasets, which can be found on their GitHub [3]. For evaluation, we extracted and utilized only the text column from each dataset. The details of the validation and test splits can be found in Table 5:

Microagressions.com: (MAgr, Breitfeller et al. (2019)) is a publicly available Tumblr blog where users can anonymously post about socially biased interactions and utterances in the wild.

Social Bias Frames: (SBF, Sap et al. (2020))is a corpus of socially biased and offensive content from various online sources.

DynaHate: (Vidgen et al., 2021) is an adversarially collected set of hate speech where human annotators create examples that an iteratively improved hate-speech classifier cannot detect.

[3]https://github.com/shallinan1/MaRCoDetoxification

Hyperparameter	Tested	Assignment
repetition penalty	[1.0, 1.2, 1.5]	1.5
α_1	[0.5, 1.0, 1.5]	1.5
α_2	[4.0, 4.25,..., 5.0]	4.75
temperature (base model)	[0.9, 1.7, 2.5]	2.5

Table 4: Hyperparameters tested and used for MaRCo on DynaHate

Dataset	# rows	
	Val	Test
MAgr	238	298
SBF	92	114
DynaHate	1,858	2,011

Table 5: Summary of Evaluation Datasets

3.3 Hyperparameters

For fine-tuning the Expert and Anti-Expert models, we used a combination of the hyperparameters listed on the original paper and their GitHub repository since the variables' names were not kept consistent. The paper did not explicitly specify the hyperparameters for train_batch_size and eval_batch_size. Instead, they exclusively mentioned the effective_batch_size. Without guidance regarding eval_batch_size, we opted to adopt the values listed for their models hosted on Hugging Face.[4]. It's worth noting that we reached out to the authors for clarification, but as of now, we have not received a response.

Besides training, the authors also searched over different hyperparameter values on each development set partition of the 3 evaluation datasets. However, it's important to note that the paper only presented a list of 4 out of 8 hyperparameters. These parameters are α_1 (weight on anti-expert for ensembling distributions during decoding), α_2 (weight on expert for ensembling distributions during decoding), and repetition penalty (how much to penalize repetition) and temperature (generation temperature), please refer to Tables 2, 3 and 4. As a result, for the parameters not detailed in the article, we utilized their default values specified in their codebase. We also have reached out to the authors to request more information about this, but as of now, we have not received a response.

3.4 Implementation

To reproduce the results found in the original MaRCo paper, we have reviewed the code repository the authors provided [3]. While the code is fairly clean, we had to make some alterations as

[4]https://huggingface.co/hallisky

Hyperparameter	Assignment
model	BART-base
dropout	0.1
encoder_layers	6
decoder_layers	6
hidden_size	768
number of GPUs	2
effective batch size	42
total steps	100,000
steps per evaluation	1,000
learning rate optimizer	AdamW
AdamW initial learning rate	2.5e-06
AdamW epsilon	1e-06
learning rate schedule	linear with no warmup
weight decay	0.0
max sequence length	180
max generation length	230
padding sequences	to max seq length

Table 6: Hyperparameters used to finetune the expert model

Hyperparameter	Assignment
model	BART-base
dropout	0.1
encoder_layers	6
decoder_layers	6
hidden_size	768
number of GPUs	1
effective batch size	32
total steps	50,000
steps per evaluation	1,000
learning rate optimizer	AdamW
AdamW initial learning rate	1.0e-6
AdamW epsilon	1e-06
learning rate schedule	linear with no warmup
weight decay	0.0
max sequence length	180
max generation length	230
padding sequences	to max seq length

Table 7: Hyperparameters used to finetune the anti-expert model

their code was not running, which we have made in our repository[5]. We have organized all essential commands for both fine-tuning and evaluation within our GitHub repository. Furthermore, the repository encompasses the datasets and rewrites utilized for the reproduction study and the supplementary content extending beyond the initially defined scope.

The code for the original implementation of MaRCo was written primarily in Python. The methodology utilizes pre-trained language models, which the authors implemented and fine-tuned with the HuggingFace Transformers library and PyTorch.

[5]https://github.com/DanielDaCosta/detoxifying-text-MaRCo

3.5 Experimental setup

We are performing experiments for our reproduction study using the USC HPC Cluster through CARC for access to adequate GPU resources. Experiments are run as jobs using the Slurm Workload Manager. Our job files can be found in our repository [5].

3.6 Computational requirements

This work was originally implemented and evaluated using the NVIDIA RTX6000 GPU. The authors found that fine-tuning the BART-base model to create their expert and anti-expert models required 40 and 2 GPU hours, respectively. The original work fine-tuned the expert model for 100,000 steps with batch size 48 (Approximately 4 epochs), and the anti-expert model for 50,000 steps with batch size 32 (11 epochs). We found it took around 0.7-0.85 GPU hours per epoch to train the anti-expert model, depending on the hardware used.

As only the NVIDIA Tesla series is available on CARC, we cannot perfectly replicate the authors' original setup. However, our initial research indicated that the Tesla V100 should be similar to the RTX6000 in performance so that similar training times could be expected. Despite this, we have found that our model training times are nearly 4x what is reported in the paper.

When fine-tuning the Anti-Expert model on CARC using a V100 GPU we observed a training time of 9.3 hours when utilizing a V100 GPU, which is approximately 4x longer than the time reported in the original paper. We also fine-tuned the same model using an A100. We reduced the time to 7.7h, still exceeding the original reported time by the authors by a factor of 3.8. After thorough checks, we confirmed that the hyperparameters we employed were identical to those specified for their model on Hugging Face. Therefore, the only plausible explanation for the discrepancy is the possibility that they utilized some different hyperparameters that were not reported.

For the Expert model, we encountered no such discrepancies. Using 2xA100s with 32GB of memory, we achieved a training time of 42h 32m, slightly exceeding the authors' reported training time of 40 hours with 2x NVIDIA RTX6000s.

To ensure a fair comparison, we fine-tuned the smaller model using the same hardware setup as previously described. For BART$_{small}$, we recorded a training time of 2h:40m for the Anti-Expert

		Validation			Test		
	Method	Toxicity (↓)	BERTScore (↑)	Fluency (↓)	Toxicity (↓)	BERTScore (↑)	Fluency (↓)
MAgr	Original	0.286	-	51.49	0.272	-	70.20
	CondBERT	0.161	0.966	104.10	0.148	0.964	88.69
	ParaGeDi	0.162	0.931	104.46	0.172	0.929	**20.78**
	MaRCo	**0.145**	0.958	43.54	**0.141**	0.954	39.10
	MaRCo (*RE*)	0.166	0.966	**40.35**	0.163	0.963	50.56
	Reproduction	0.181	**0.970**	45.19	0.177	**0.969**	39.91
SBF	Original	0.351	-	58.46	0.344	-	88.79
	CondBERT	0.202	**0.961**	69.51	0.190	**0.961**	131.12
	ParaGeDi	0.186	0.921	79.88	0.192	0.923	99.96
	MaRCo	**0.176**	0.947	54.86	**0.186**	0.946	48.75
	MaRCo (*RE*)	0.198	0.954	**46.73**	0.213	0.952	69.77
	Reproduction	0.234	**0.961**	48.81	0.221	0.952	**43.36**
Dyna Hate	Original	0.563	-	205.73	0.578	-	220.42
	CondBERT	0.288	0.954	190.51	0.293	0.950	200.20
	ParaGeDi	0.332	0.918	217.78	0.323	0.912	240.17
	MaRCo	**0.274**	0.939	**110.50**	**0.277**	0.936	**128.84**
	MaRCo (*RE*)	0.305	**0.951**	125.20	0.302	0.948	145.19
	Reproduction	0.337	0.955	120.07	0.339	**0.951**	148.19

Table 8: Evaluations on detoxified generations on MAgr, SBF, and DynaHate for MaRCo, ParaGeDi, CondBERT, our fine-tuned version labeled as *Reproduction* and Marco (*RE*) – our re-evaluation of the author's original model. The best score for each column/dataset is bolded, and the second best is underlined.

model and 16h:42m for the Expert.

3.7 Automatic Evaluation Metrics

Similar to Hallinan et al. (2023) and prior work (Liu et al. (2021), Ma et al. (2020)), we will evaluate the performance of MaRCo and the baseline models using the following automatic metrics.

Toxicity: average score of rewrites from the Perspective API, a publicly accessible toxicity classifier trained on the Jigsaw dataset. The API generates scores ranging from 0 to 1, with higher values indicating a higher likelihood of a reader perceiving the comment as toxic.

Fluency: Measured by computing the perplexity of rewrites using GPT-2 XL (Radford et al., 2019). This assumes GPT-2 XL to be the "fluent" baseline and measures the degree to which the rewrites diverge.

Meaning Similarity: This measurement is determined by the difference in BERTScore between the input and its corresponding rewrite, as proposed in (Zhang et al., 2019). The authors use RoBERTa-large for this purpose, which has 354M parameters (Liu et al., 2019).

Detoxify: We add metric measuring toxicity that is version-independent called detoxify (Hanu and Unitary team, 2020). Detoxify is designed to evaluate and quantify toxicity in language, particularly in online interactions and digital content. We use the "original" model that employs a BERT-based language model trained on data from the Toxic Comment Classification Challenge (cjadams, 2017). This model assigns scores for 6 different categories of hate speech and toxic languages: toxicity (which is used in this study), severe toxicity, obscene, threat, insult, identity attack, and sexually explicit.

4 Results

4.1 Reproducibility

To compare our obtained results with those reported by the authors, we refer to Table 8 for a comprehensive overview. In the context of Toxicity, our reproduction study consistently yielded lower results than what was reported. This discrepancy persisted across all evaluations conducted on both the Validation and Test sets for each dataset. Such findings are directed contraction to the claim 1. To validate this finding, we also evaluate the models provided by the authors on Hugging Face to delve into this matter. Despite MaRCo (*RE*) yielding results worse than reported, our reproduction model still performed worse. This discrepancy between our re-evaluation and their original model further strengthens the hypothesis that hyperparameters or other critical details are missing from the original paper.

Regarding fluency, our reproduction model demonstrated superior fluency compared to the

		Test			
	Method (Expert × Anti Expert)	Toxicity (↓)	BERTScore (↑)	Fluency (↓)	Detoxify (↓)
MAgr	$\text{BART}_{base} \times \text{BART}_{base}$	0.177	**0.969**	**39.91**	0.161
	$\text{BART}_{small} \times \text{BART}_{small}$	0.217	0.966	57.56	0.120
	$\text{BART}_{base} \times \text{BART}_{small}$	0.224	0.887	114.02	0.143
	$\text{BART}_{small} \times \text{BART}_{base}$	0.206	0.953	81.59	0.195
	LLM Prompting	**0.140**	0.929	63.40	**0.055**
SBF	$\text{BART}_{base} \times \text{BART}_{base}$	0.221	**0.952**	43.36	0.175
	$\text{BART}_{small} \times \text{BART}_{small}$	0.258	0.892	114.24	0.183
	$\text{BART}_{base} \times \text{BART}_{small}$	0.331	0.910	156.34	0.208
	$\text{BART}_{small} \times \text{BART}_{base}$	0.220	0.899	176.91	0.139
	LLM Prompting	**0.159**	0.926	**37.64**	**0.057**
Dyna Hate	$\text{BART}_{base} \times \text{BART}_{base}$	0.339	0.951	148.19	0.293
	$\text{BART}_{small} \times \text{BART}_{small}$	0.440	**0.962**	190.44	0.371
	$\text{BART}_{base} \times \text{BART}_{small}$	0.434	0.926	222.77	0.347
	$\text{BART}_{small} \times \text{BART}_{base}$	0.414	0.945	248.93	0.350
	LLM Prompting	**0.274**	0.916	**85.45**	**0.117**

Table 9: Experiments beyond the original paper: Evaluations were run Evaluation metrics scores on the cross combination of expert and anti-expert models using the original BART-base and BART-small models; LLM (Mistral-7B) prompting and our *Reproduction* labeled here as $\text{BART}_{base} \times \text{BART}_{base}$. The best score for each column/dataset is bolded, and the second best is underlined. Validation results were omitted for brevity.

other models, excluding the original MaRCo model. As a result, claim 2 is proven to be correct. The outcome underscores MaRCo's proficiency in generating meaningful sentences, highlighting its ability to replace toxicity tokens and create coherent and cohesive sentences. Our model outperformed other models, except for the original MaRCo Model, when assessing the BERT Score. This suggests that the rewriting process successfully retained the similarity of sentences while generating less toxic tokens, thus substantiating 3.

4.2 Additional results not present in the original paper

We present the results of our additional claims tested in table 9.

Among the MaRCo-based approaches, the original model (BART-base expert/anti-expert) generally performed best for toxicity reduction with the Perspective API scores. However, there is significantly more variability in the rankings with the Detoxify metric. Based on these results, we would still recommend the original model with 2 versions of BART-base, as the Perspective API is more sophisticated than Detoxify, but in settings with limited computing available, this seems to indicate that using BART-small is a potentially viable approach. Regarding efficiency, the training time for BART_{small} Expert and Anti-Expert is approximately 19h20m, marking a significant 62% reduction in duration compared to that of BART_{base}.

Notably, LLM prompting performs best in toxicity reduction across both the Perspective API and Detoxify metrics for all datasets. However, the rewrites for this method tend to perform poorly in terms of BERTScore, indicating that the rewrites diverge more heavily from the original text when compared with the other methods tested. In addition, qualitative evaluation of the rewrites shows that while they tend to rephrase the original text in a less "aggressive" way, the original toxic meaning remains. Given this, we feel that while this method seems to perform well at toxicity reduction, it is not currently a viable approach for the task.

5 Discussion

5.1 What was easy

The code from the original paper was easily accessible on GitHub, which had a significant positive impact on reproduction by allowing for a quick and easy start to our efforts. However, while the code was well documented, some changes were required to make it fully functional. Initially, we had to reorganize the file structure and resolve issues with the `environment.yml` file that was provided. Additionally, we rectified errors associated with incorrectly passed variables to functions and identified missing variables that needed inclusion.

The methodology is also well described in the original paper, which allowed us to easily interpret and understand the code.

5.2 What was difficult

Our main challenge in reproduction came from missing or under-specified hyperparameters. While most hyperparameters were well specified, some parameters were not provided. In the fine-tuning section, the original authors omitted the parameter eval batch size, as they solely provided the effective batch size. As previously discussed, we resolved this by adopting the parameter values from their models on Hugging Face. In evaluation, crucial parameters for the decoding process, such as top p (nucleus sampling), top k, and batch size (how many sequences to generate at once), were not provided by the original authors. As a result, we decided to use their default values for all experiments.

While we attempted to specify these by contacting the authors, there were issues with their responsiveness, which made this difficult. While we were eventually able to resolve the issue for training hyperparameters looking at the `config.json` files for the author's original models on Hugging Face[6], we are still attempting to replicate the parameters for the authors' evaluations.

In addition, we faced issues matching the GPU hardware used in the original work. While the authors trained their models with the NVIDIA RTX6000, our experiments were conducted on CARC, which does not have this model. As such, we initially trained our models on the V100 GPU, which our research indicated to be a comparable GPU. However, this led to nearly 8X the original training times reported by the authors. We eventually upgraded to the A100, which yielded similar training time for the anti-expert model but still had a nearly 4X slowdown for the expert model. This indicates that there were still issues with the hyperparameters we used.

5.3 Recommendations for reproducibility

As previously mentioned, the code was not fully functional out-of-the-box. While this was a relatively minor issue, fixing it would improve the reproducibility of the original work. In addition, the documentation should include the commands used to run the original experiments, which are not currently available.

Hyperparameters should also be more clearly provided, particularly in the context of arguments used to run the experiments.

[6]Expert Model, Anti-expert Model

The authors also use the Perspective API to measure toxicity in their experiments which is often updated. As such, outputs from the API are not deterministic and may change over time. This is a significant issue in the reproducibility of the work. In addition, the Perspective API is closed source and its inner workings are not readily documented. To remedy this, more open-source metrics should be used, such as the Detoxify metric we use in our extension work.

6 Communication with original authors

While we had some brief correspondence with the authors of the original paper, our communications did not prove helpful in aiding our reproduction efforts. We initially faced issues contacting the first author, as he had graduated in June 2023, but were eventually able to reach him through the second author. However, while we sent several emails with questions about our issues in reproduction, responses promised to follow up later but would be followed by long periods of silence without answering our questions. While we were able to answer some of our questions by investigating the code, we were not able to figure out everything on our own, which likely contributed to some of our mismatched results.

This pattern of delayed/incomplete responses and its effects on our reproduction efforts underscores the importance of communication for reproducibility and successful research.

References

Luke Breitfeller, Emily Ahn, David Jurgens, and Yulia Tsvetkov. 2019. Finding microaggressions in the wild: A case for locating elusive phenomena in social media posts. 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)*, pages 1664–1674, Hong Kong, China. Association for Computational Linguistics.

Julia Elliott Lucas Dixon Mark McDonald nithum Will Cukierski cjadams, Jeffrey Sorensen. 2017. Toxic comment classification challenge.

David Dale, Anton Voronov, Daryna Dementieva, Varvara Logacheva, Olga Kozlova, Nikita Semenov, and Alexander Panchenko. 2021. Text detoxification using large pre-trained neural models. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 7979–7996, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Skyler Hallinan, Alisa Liu, Yejin Choi, and Maarten Sap. 2023. Detoxifying text with marco: Controllable revision with experts and anti-experts.

Laura Hanu and Unitary team. 2020. Detoxify. Github. https://github.com/unitaryai/detoxify.

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.

Alisa Liu, Maarten Sap, Ximing Lu, Swabha Swayamdipta, Chandra Bhagavatula, Noah A. Smith, and Yejin Choi. 2021. DExperts: Decoding-time controlled text generation with experts and anti-experts. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 6691–6706, Online. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach.

Xinyao Ma, Maarten Sap, Hannah Rashkin, and Yejin Choi. 2020. PowerTransformer: Unsupervised controllable revision for biased language correction. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 7426–7441, Online. Association for Computational Linguistics.

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.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5477–5490, Online. Association for Computational Linguistics.

Bertie Vidgen, Tristan Thrush, Zeerak Waseem, and Douwe Kiela. 2021. Learning from the worst: Dynamically generated datasets to improve online hate detection.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q. Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with BERT. *CoRR*, abs/1904.09675.