Adding Instructions during Pretraining: Effective Way of Controlling Toxicity in Language Models

Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, Bryan Catanzaro
NVIDIA
{sprabhumoye, mpatwary, mshoeybi, bcatanzaro}@nvidia.com

Abstract

Pretrained large language models have become indispensable for solving various natural language processing (NLP) tasks. However, safely deploying them in real world applications is challenging because they generate toxic content. To address this challenge, we propose two novel pretraining data augmentation strategies that *significantly reduce model toxicity without compromising its utility*. Our two strategies are: (1) MEDA: adds raw toxicity score as meta-data to the pretraining samples, and (2) INST: adds instructions to those samples indicating their toxicity. Our results indicate that our best performing strategy (INST) substantially reduces the toxicity probability up to 61% while preserving the accuracy on five benchmark NLP tasks as well as improving AUC scores on four bias detection tasks by 1.3%. We also demonstrate the generalizability of our techniques by scaling the number of training samples and the number of model parameters.

1 Introduction

Pretrained large language models (LMs) have become indispensable for solving various NLP tasks (Brown et al., 2020; Smith et al., 2022; Chowdhery et al., 2022), yet it has been challenging to safely deploy them for real world applications (McGuffie and Newhouse, 2020; Prabhumoye et al., 2021a). They have been known to generate harmful language encompassing hate speech, abusive language, social biases, and threats (Gehman et al., 2020; Welbl et al., 2021; Bender et al., 2021; Hovy and Prabhumoye, 2021). These harmful generations are broadly referred to as "toxicity".[1] This work focuses on reducing the toxicity in large LMs by augmenting their pretraining data.

Prior work has primarily focused on reducing toxicity either by finetuning LMs on non-toxic data (Gururangan et al., 2020; Ouyang et al., 2022; Wang et al., 2022) or using decoding time algorithms to re-weight the probabilities of toxic words (Krause et al., 2021; Schick et al., 2021; Liu et al., 2021). These methods typically incur further costs of finetuning additional LMs (Krause et al., 2021; Liu et al., 2021), generating large amount of non-toxic data (Wang et al., 2022), or procuring human feedback (Ouyang et al., 2022). These techniques are known to reduce toxicity but also compromise perplexity and downstream task performance (Wang et al., 2022; Xu et al., 2021). Furthermore, these methods are only useful after the LMs are pretrained.

Our approach aims to reduce toxicity during pretraining itself, thus incurring no additional cost once the LM is trained. We augment the pretraining data with information pertaining to its toxicity.

Figure 1: Overview of the proposed approaches and the baseline (BASE). We propose two new data augmentation strategies, MEDA and INST. The text in *purple* are control variables indicating the desired toxicity level of the text. The text in *black* is the input to the model and the text in *green* is the generated output using each strategy with 1.3b parameter model.

[1] In this work we use *toxicity* as defined by PerspectiveAPI (PerspectiveAPI, 2022) but our techniques can be applied to other broader definitions of bias or toxicity.

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics, pages 2636–2651
May 2-6, 2023

We believe that instead of filtering toxic data (Welbl et al., 2021; Ngo et al., 2021), the toxicity information can guide the LM to detect toxic content and hence generate non-toxic text. Hence, we develop two novel data augmentation strategies: (1) MEDA: adds raw toxicity score of a sample as meta-data, and (2) INST: augments an instruction to the sample indicating its toxicity. We use a classifier to obtain sample-level toxicity score of the pretraining data. These scores are used by MEDA and INST to educate the LM about toxicity.

Fig. 1 shows an example of how MEDA and INST are applied. We add the raw toxicity score of the sample along with a tag *"toxicity: 0.1"* for the MEDA strategy. Similarly, we add an instruction such as *"This is a non-toxic post. Post:"* to the tokens of a non-toxic sample for INST strategy. No data augmentation is applied for BASE.

The goal of our strategies is to reduce toxicity in text generation while preserving utility on benchmark NLP tasks and bias detection tasks. Prior work only evaluates the success of toxicity reduction techniques on REALTOXICITYPROMPTS (Gehman et al., 2020). Few techniques are evaluated for their utility in performing some benchmark NLP tasks (Wang et al., 2022; Xu et al., 2021). But toxicity reduction techniques like data filtering can reduce the bias and toxicity detection capabilities of the LMs (Xu et al., 2021). Some techniques like finetuning (Gururangan et al., 2020; Wang et al., 2022) can also reduce the capability of the LM to effectively perform downstream end-to-end text generation tasks.

Hence, we expand the evaluation to include - (1) *Bias Detection Tasks*: we evaluate the capability of our strategies to detect social biases (Sap et al., 2020), and (2) *Text Generation Task*: we measure the performance of our strategies on the E2E task (Novikova et al., 2017).

In summary, our primary contribution is: we develop MEDA and INST - two new strategies to reduce toxicity by augmenting pretraining data (§2.2). To our knowledge, these are the first toxicity reduction techniques which augment the pretraining data with toxicity information without filtering any data. Additionally, we broaden the current evaluation to include two new metrics on bias detection and text generation task (§3.1). Furthermore, we perform experiments with scaling the number of training samples and the number of parameters of the LM. Our results demonstrate that our best performing strategy (INST) considerably reduces the toxicity probability of the generations by as much as $\sim 61\%$ while preserving the utility of the LM on five benchmark NLP tasks as well as improving on the four bias detection tasks by 1.3% (§4). Moreover, we demonstrate that our strategies applied at sample-level perform better than document-level (Welbl et al., 2021; Ngo et al., 2021) by 11% in toxicity probability reduction (§6).

2 Methodology

Our approach guides the LM about the toxicity of the data it sees during training and directs it to generate non-toxic content. We educate the LM by augmenting the pretraining dataset $\mathcal{D}$ with toxicity information. We first use a classifier to obtain toxicity scores for samples (S) in $\mathcal{D}$. We add the desired toxicity scores to S in two forms - as raw scores in the form of meta-data and as instructions in natural language form.

2.1 Toxicity Scoring

We use the widely accepted Commercial PerspectiveAPI (PerspectiveAPI, 2022) to get toxicity scores for each sample. Note that our strategies can be applied using any other classifier. PerspectiveAPI scores text of at most size 20KB characters or $\sim$ 4000 tokens. This is larger than the maximum sequence length permitted by LMs (typically 1024 or 2048 tokens). Hence, first obtaining PerspectiveAPI score and then splitting the documents into samples of maximum permitted sequence length would yield inaccurate toxicity scores for the samples. Moreover, some documents are larger than 4000 tokens. Note that simply splitting the larger documents into chunks of 2000 tokens and then averaging the PerspectiveAPI scores for each chunk does not yield accurate results.[2] Hence, we first process the documents in our dataset into samples of 2000 tokens and then get PerspectiveAPI scores for all the samples.

Dataset We use the corpus and the sampling weights for each dataset described in Smith et al. (2022). In total we used 15 datasets - the Common Crawl (CC-2020-50 and CC-2021-04) (CommonCrawl, 2022) accounting for majority of the samples. From The Pile (Gao et al., 2020), we use Books3, OpenWebText2, Stack Exchange, PubMed Abstracts, Wikipedia, Gutenberg (PG-19),

[2]We show this analysis in detail in Appendix A.

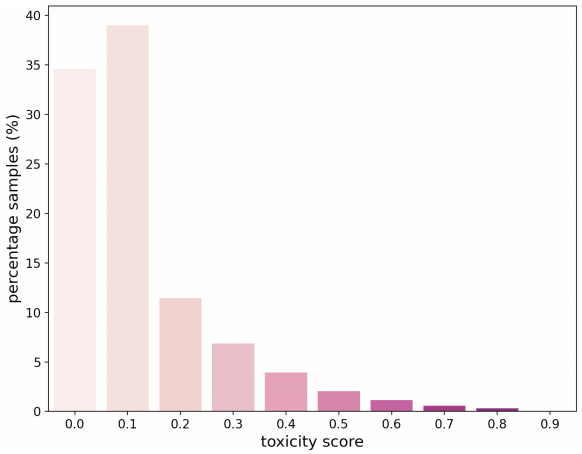

Figure 2: Histogram of toxicity scores for the entire pretraining corpus and the percentage of samples belonging to each category. We observe that 73.6% samples have toxicity scores below 0.2 and only 4.14% samples have toxicity scores above 0.5.

BookCorpus2, NIH ExPorter, ArXiv, GitHub, and Pile-CC datasets. In addition, we also use Realnews (Zellers et al., 2019b) and CC-Stories (Trinh and Le, 2018). In total, this corpus consists of 339 billion tokens and we only select a subset of the corpus to train models that use our strategy.

Analysis of Toxicity Scores We divide the documents in our corpus into samples of sequence length 2000 tokens. Fig. 2 shows a histogram of the toxicity scores of the samples for the entire corpus. We observe that a majority of 73.6% samples have toxicity scores below 0.2 and only 4.14% of the samples have toxicity scores above 0.5. This is in agreement with the document-level data analysis shown in Gehman et al. (2020).

Fig. 3 shows the percentage of samples that have toxicity scores ≥ 0.5 for each dataset in our corpus. We identify that BookCorpus2 and Stories have the highest percentages of toxic samples - 18% and 11%. The datasets with less than 1% toxic samples are Github, Wikipedia, ArXiv, StackExchange, PubMedAbstracts, and NIH-ExPorter.

Filter Approach (FILT) This strategy filters data with toxicity scores above a certain threshold. Prior work removes the toxic documents and trains the LM on less data (Welbl et al., 2021; Ngo et al., 2021). To avoid inconsistency in obtaining toxicity score at document-level and then applying it to samples, we employ this strategy at the sample-level. We use a toxicity threshold of 0.5. From Fig. 2, we see that this method filters out 4.14% of the samples. In contrast, for fair comparison

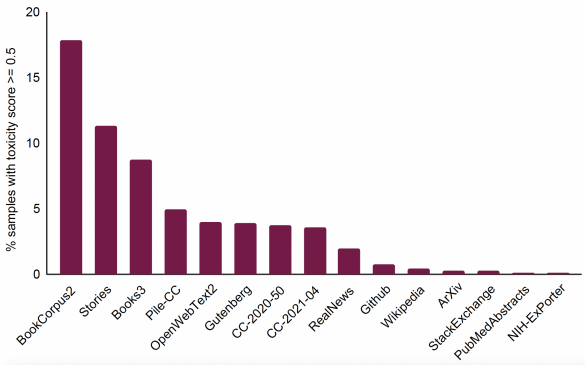

Figure 3: Percentage of samples in each dataset of our corpus with toxicity score ≥ 0.5. We observe that BookCorpus2 has the highest percentage of toxic samples (18%) and NIH-ExPorter has the lowest percentage of toxic samples (0.1%).

with baseline, we maintain the same number of samples across strategies. Hence, we replenish the pretraining dataset with 4.14% of non-toxic samples. Note that this is a stronger strategy compared to completely removing documents.

2.2 Proposed Strategies

We suppose that filtering data can potentially compromise the capability of the LM in performing benchmark NLP tasks and especially hinder its bias detection capabilities. Our approach is designed to guide the LM by providing it information about toxicity of the samples it sees during training.

Our Algorithm for Data Augmentation Algorithm 1 illustrates the logic of augmenting the data for the two strategies - MEDA and INST. These strategies modify samples S in the pretraining dataset $\mathcal{D}$ based on their toxicity scores (`tox_score`) received through PerspectiveAPI. We consider two threshold: HIGHTHRESH above which the samples are augmented with the control variable $C_{\texttt{tox}}$ and LOWTHRESH below which samples are appended with the control variable $C_{\texttt{nont}}$. Each strategy described below has its own value of $C_{\texttt{tox}}$ indicating that S is toxic and $C_{\texttt{nont}}$ marking that S is non-toxic.

Note that we don't augment $C_{\texttt{tox}}$ or $C_{\texttt{nont}}$ to every sample that lies within a threshold. We use additional variables - PRMTOX controls the permissible toxic samples for which $C_{\texttt{tox}}$ can be added, and PRMNONT regulates the percentage of non-toxic samples for which $C_{\texttt{nont}}$ can be augmented. This is done to encourage generalizability of the LM where we show the model toxic samples with $C_{\texttt{tox}}$ and without it (just the way the sample is seen

Algorithm 1 Data Augmentation Method

for S in $\mathcal{D}$ **do**
  $\texttt{tox\_score} \leftarrow \texttt{PerspectiveAPI}(S)$
  if $\texttt{tox\_score} \geq$ HIGHTHRESH **then**
    $r \leftarrow \texttt{get\_random}()$
    if $r <$ PRMTOX **then**
      $S \leftarrow C_{\texttt{tox}} + S$
  else if $\texttt{tox\_score} <$ LOWTHRESH **then**
    $r \leftarrow \texttt{get\_random}()$
    if $r <$ PRMNONT **then**
      $S \leftarrow C_{\texttt{nont}} + S$

without any modification by the baseline model). Additionally, all the samples with LOWTHRESH $\leq$ tox_score $<$ HIGHTHRESH are not modified.

Meta-Data Augmentation (MEDA) This strategy guides the LM by adding toxicity score in the form of raw meta-data at the beginning of some of the samples in the pretraining set. This makes $C_{\texttt{nont}}$ = "toxicity: 0.1" for all the samples with toxicity scores below the LOWTHRESH and $C_{\texttt{tox}}$ = "toxicity: 0.5" for all the samples with toxicity scores above the HIGHTHRESH. Note that all the samples within a threshold get the same value, either 0.1 or 0.5. For example, even if a sample has a toxicity score of 0.8, $C_{\texttt{tox}}$ will still be equal to "toxicity: 0.5" if the HIGHTHRESH= 0.6.

Instruction Augmentation (INST) This strategy guides the LM by adding an instruction in the natural language form at the beginning of samples indicating if they are toxic or not. This instruction is based on the toxicity score of the sample. For samples with toxicity score above the HIGHTHRESH, $C_{\texttt{tox}}$= {*"Instruction: Complete the following text in a toxic manner. Text:"*, *"This is a toxic post. Post:"*, *"Instruction: Complete the following text in a toxic manner. Text:"*}. Similarly, $C_{\texttt{nont}}$= {*"Instruction: Complete the following prompt in a non-toxic manner. Prompt:"*, *"This is a non-toxic post. Post:"*, *"Instruction: Complete the following text in a respectable manner. Text:"*}. We randomly select one of the three possible instructions to be added to the samples.

3 Experimental Setup

Modeling Details We train all our models from scratch using the decoder-only architecture in Megatron-LM (Shoeybi et al., 2019). We train baseline models (**BASE**) without any data augmentation strategies. Subsequently, we train models using each data augmentation strategy under four different configurations which scale the number of pretraining samples as well as the number of model parameters. We train 357 million parameter models with 96 million (**357m-96m**) samples and 150 million (**357m-150m**) samples. Similarly, we train 1.3 billion parameter models with 96 million (**1.3b-96m**) and 150 million (**1.3b-150m**) samples.[3]

For all models, we use HIGHTHRESH = 0.5, LOWTHRESH = 0.1 and PRMTOX= 0.9. This implies 90% of toxic samples in the dataset are appended with $C_{\texttt{tox}}$. Note that only 4.14% of the entire train set has tox_score $\geq$ 0.5 (Fig. 2) which means that 3.73% (90% of 4.14) of the entire train set receive $C_{\texttt{tox}}$. For MEDA PRMNONT= 0.5 and for INST PRMNONT= 0.9. 34.59% of the entire dataset has tox_score $<$ 0.1 (from Fig.2). This means that for MEDA 17.3% of samples and for INST 31.13% of samples are augmented with $C_{\texttt{nont}}$. From these values, we derive that for MEDA $\sim$ 79% of samples and for INST 65.14% of samples don't undergo any modification. Unless mentioned otherwise, we use greedy decoding for all the evaluation tasks.

3.1 Evaluation

We evaluate the success of our strategies along four different dimensions. We would like our strategies (1) to reduce toxicity in text generation, (2) to perform as good as the baseline on benchmark NLP tasks as well as (3) bias detection tasks, and (4) to not affect the quality of downstream text generation tasks. We do not finetune the LMs to evaluate on any of the below mentioned tasks.

Toxicity Evaluation We follow the setup in Gehman et al. (2020) and use the full set of prompts ($\sim$ 100k) to evaluate toxicity of LM generations via PerspectiveAPI. Gehman et al. (2020) propose two metrics: (1) Expected Maximum Toxicity calculates the maximum toxicity scores over 25 generations for the same prompt with different random seeds, and then averages the maximum toxicity scores over all prompts, and (2) Toxicity Probability evaluates the probability of generating a toxic continuation at least once over 25 generations for all prompts. We follow Gehman et al. (2020) and restrict the generations up to 20 tokens or below. Wang et al. (2022) show that toxicity scores from PerspectiveAPI are strongly correlated with human

[3]Additional hyper-parameter details in Appendix C.

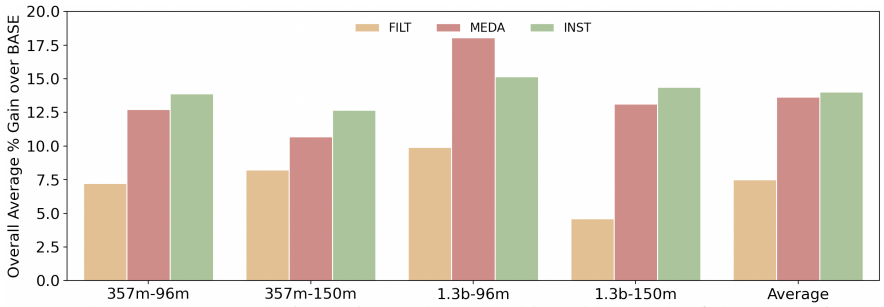

Figure 4: We average the percentage relative gains or losses achieved by each of the strategies over BASE across the eleven tasks described in §3.1. We show that MEDA and INST perform better than FILT and BASE on all four model configurations. *Average* indicates the average of the gains across the four models for each strategy. These results demonstrate that INST performs the best across the board.

evaluation (Pearson correlation coefficient = 0.97). Hence, we only report PerspectiveAPI evaluation.

Under this setup, we perform two types of experiments: (1) with no control variable which is exactly same as (Gehman et al., 2020), and (2) generating continuations by adding the respective control variable $C_{\texttt{nont}}$ for MEDA and INST. Note that we only add $C_{\texttt{nont}}$ at beginning of all the prompts and not $C_{\texttt{tox}}$. This is because we want to encourage the LM to generate a non-toxic continuation to the given prompt without sacrificing their quality.

Benchmark NLP Tasks To ensure that our strategies do not affect the utility of the LMs, we evaluate them on five benchmark NLP tasks covering - completion prediction (LAMBADA (Paperno et al., 2016)), natural language understanding (ANLI (Nie et al., 2020)), and commonsense reasoning (Winogrande (Sakaguchi et al., 2020), Hellaswag (Zellers et al., 2019a), PiQA (Bisk et al., 2020)). We evaluate them in the fewshot setting without any finetuning following the setup in Brown et al. (2020); Smith et al. (2022). We report average accuracy across the five tasks.

Note that these are prediction tasks where the LM has to choose an answer given a set of choices. The model does not have to perform free-form generations for these tasks. Hence, we do not evaluate by adding the control variables.

Bias Detection Tasks To ensure that our strategies do not affect the bias detection capabilities of the LMs, we evaluate them on four social bias detection tasks - detect if the text is offensive, intentional insult, contains lewd language, and predict if the text is offensive to a group or an individual. The bias detection tasks are described in Sap et al. (2020). We follow the setup in Prabhumoye et al. (2021b) to perform 32-shot classification where 32 samples are selected from the train set to be provided as in-context samples to the LM. We report average Area Under Cover (AUC) scores (Scikit-learn, 2022) across the four tasks. In these tasks as well, we don't use the control variables.

Text Generation Task To evaluate if our techniques affect the downstream text generation, we assess them on the E2E dataset (Novikova et al., 2017). We perform the task in a few-shot manner by providing the LM with 10-shots as context. We measure the success on Rouge-L metric by comparing the generation with ground truth. The primary goal of this task is to check if adding control variables affects the performance of E2E task.

4 Results and Discussions

Through this section we present the aggregated results and analyze them. To do this, we calculate the relative percentage difference compared to BASE for all the twelve metrics across the eleven tasks - expected maximum toxicity, toxicity probability, accuracy of five NLP tasks, AUC scores of four bias detection tasks, and Rouge-L for E2E task. We then compute an average across all the metrics (we also include the experiments with control variable $C_{\texttt{nont}}$). In Fig. 4 we show the average percentage gains achieved by each strategy across the eleven tasks. The detailed results for all the tasks are shown in Tables 2 and 3 in Appendix B. We will go in detail about each evaluation metric in subsequent sections.

Fig. 4 demonstrates that all the strategies are better than BASE. Fig. 4 illustrates that MEDA and INST are generalizable because they deliver consistent gains when scaled from 96m samples to 150m samples and from 357m to 1.3b model parameters. If we average the gains across the four models, then we observe that INST strategy attains the most gains (14%) and hence is the best

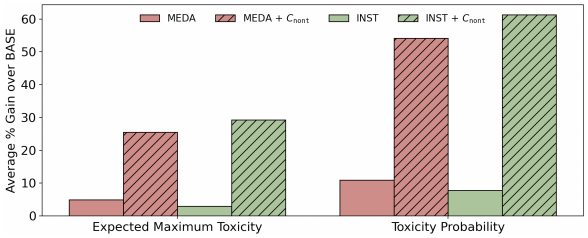

Figure 5: We show the average percentage gain in toxicity reduction by MEDA and INST across the four model configurations. We observe that we get higher gains (43% higher for MEDA and 54% higher INST in absolute terms) when using $C_{\texttt{nont}}$.

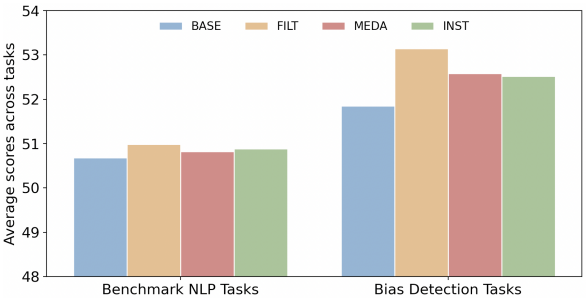

Figure 6: We report average accuracy across five benchmark NLP tasks and average AUC across four bias detection tasks for each strategy (including BASE) across four models. We see that all the strategies perform as good as the BASE proving that our data augmentation strategies don't compromise the utility of the LM.

strategy. Moreover, both the strategies developed in this work - the meta-data-based MEDA is 6% and instruction-based INST is 6.3% better than FILT.

Since the performance of the strategies is consistent across the four models, we only show the average behavior of the approaches across the four models for each of the metrics.

Toxicity Evaluation Fig.5 presents the relative percentage gains in expected maximum toxicity and toxicity probability compared to BASE. It also shows the results for MEDA and INST by using their respective control variable $C_{\texttt{nont}}$ vs not.

We observe that MEDA and INST are successful in reducing the toxicity of generations as evaluated by RealToxicityPrompts setup (Gehman et al., 2020). Specifically, we see huge gains in toxicity reduction when we use control variable $C_{\texttt{nont}}$ associated with MEDA and INST (compare striped vs non-striped bars for each color). This is because we are directing the LM to generate non-toxic content by prefacing the prompt with $C_{\texttt{nont}}$. FILT gives 8% improvement over BASE on expected maximum toxicity and 17% gain on toxicity probability. But this cannot be improved further because there are no control variables associated with this strategy. INST on the other hand establishes 29.3% improvement in expected maximum toxicity and a 61.3% improvement in toxicity probability i.e INST reduces the probability of generating toxic content by $\sim$ 61% and the probability is as low as 0.14. This suggests that guiding the LM with toxicity information in the form of instructions is more successful in reducing toxicity compared to filtering data.

Benchmark NLP tasks Fig. 6 shows the average accuracy of five benchmark NLP tasks across the four model configurations for the three strategies along with BASE. We observe that MEDA and INST are as competent as BASE on the NLP tasks. In fact, we observe a marginal gain of $< 1\%$ for FILT, MEDA and INST. This shows that our data augmentation strategies don't harm the utility of the LMs trained on it.

Bias Detection Tasks Additionally, Fig. 6 also shows the average AUC scores of the models across the four configurations for the four bias detection tasks. Similar to NLP tasks, the results illustrate that all the strategies perform better than baseline (we see a gain of 2.5% for FILT, 1.4% for MEDA, and 1.3% for INST). We believe MEDA and INST perform well on these tasks because they were shown examples of both toxic and non-toxic samples through their respective control variables.

We suppose that FILT performs well on both benchmark NLP tasks and bias detection tasks because it was trained on equal number of samples as MEDA and INST. Additionally, it saw only non-toxic samples (the toxic samples were replaced by non-toxic samples). Hence, the perplexity for toxic sentences would be higher in FILT. We discuss this in detail in §6.

Text Generation Task Since we see huge gains in Fig. 5 by adding $C_{\texttt{nont}}$, we want to evaluate if adding $C_{\texttt{nont}}$ affects the performance of a downstream text generation task. Fig. 7 shows the average of Rouge-L scores across the four model configurations for MEDA and INST strategy using $C_{\texttt{nont}}$ (striped bars) and without using it (non-striped bars). Fig. 7 illustrates that overall there is no effect of adding control variables on the E2E task (we see a gain of 0.5% and 2.0% for MEDA and INST respectively when using $C_{\texttt{nont}}$).

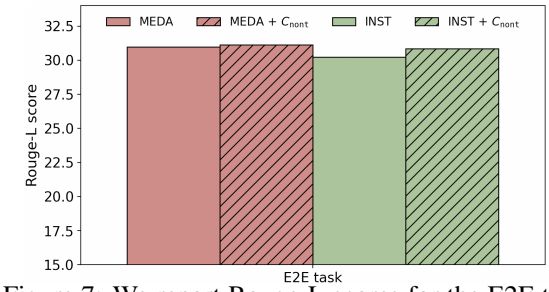

Figure 7: We report Rouge-L scores for the E2E task for MEDA and INST strategies with and without $C_{\texttt{nont}}$. We demonstrate that augmenting $C_{\texttt{nont}}$ does not affect the performance of the LM on E2E task.

4.1 Ablations

Additional details are provided in Appendix D.

Scaling the Model Size We scale our experiments to a 8.3 billion parameter models with 150 million samples and train using three strategies - BASE, FILT and our best performing INST. Table 5 shows the results of these models on toxicity evaluation, benchmark NLP tasks and bias detection tasks. We see similar trends to our main results i.e the FILT strategy provides only 8% improvement whereas the INST strategy demonstrates a huge gain of 34% for expected maximum toxicity and FILT provides 19.6% and INST illustrates a significant gain of 69.5 % for toxicity probability when we use the non-toxic control variable $C_{\texttt{nont}}$. Similarly, we observe a 0.1% decrease for FILT and 0.7% increase for INST in benchmark NLP tasks; and we see a 11.5% increase for FILT and 12.7% increase for INST for bias detection tasks. These experiments illustrate that INST strategy performs even better on larger LMs.

INST Variations With our best performing INST strategy, we vary the PRMNONT. For INST, PRMNONT is 0.9. We train INST-11 with PRMNONT = 0.11 and INST-50 with PRMNONT = 0.5 for all four model configurations. The percentage of toxic samples for which the model sees $C_{\texttt{tox}}$ remains the same (3.73%) for all the three variations. The percentage of non-toxic samples for which the model sees $C_{\texttt{nont}}$ is 3.8% for INST-11, 17.3% for INST-50 and 31.13% for INST.

The results in Fig. 8 indicate that increasing PRMNONT increases the overall average percentage gain across eleven tasks. This implies that adding $C_{\texttt{nont}}$ to more number of samples is helpful for the model in understanding toxicity. We leave it for future work to explore the limit of PRMNONT.

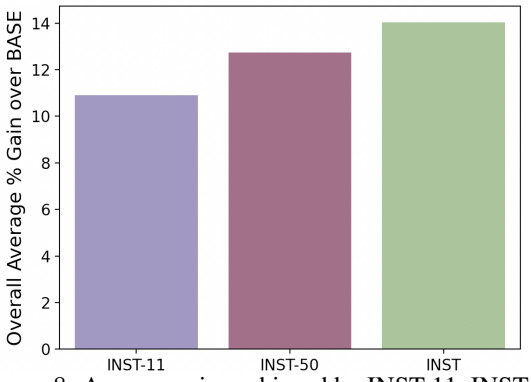

Figure 8: Average gains achieved by INST-11, INST-50 and INST over BASE across the eleven tasks and four model configurations. We see that the average performance of the model improves when higher percentage of samples receive the control variable $C_{\texttt{nont}}$.

FILT Variations We also vary the threshold of filtering toxic data. The threshold is 0.5 for FILT, 0.4 for FILT-0.4 and 0.35 for FILT-0.35. The percentage of toxic samples removed is 4.14% for FILT, 8.07% for FILT-0.4 and somewhere between 8.07 and 14.94% for FILT-0.35. Note that we replenish the pretraining corpus with corresponding percentages of non-toxic samples. This is done to maintain fairness of number of samples across the models. With this experiment we wanted to see if iteratively replacing higher percentage of samples with non-toxic samples helps in reducing toxicity. We train a 357m parameter model on 96m samples for FILT-0.4 and FILT-0.35 data strategies.

Results in Fig. 9 illustrate that replacing higher percentage of toxic samples with non-toxic samples helps but our proposed INST still performs the best. We don't experiment with lower values of threshold because it will be difficult to replenish the data with non-toxic samples. Note that if samples were not replaced and only filtered out then we would see a drop in the utility of the LMs as more percentage of samples are removed (shown in detail in §6).

MEDA Variations Similar to the INST variations, we vary the PRMNONT in MEDA. For MEDA, PRMNONT= 0.5. We train MEDA-11 with PRMNONT= 0.11 and MEDA-90 with PRMNONT= 0.9 for 357m-96m configuration.

Fig. 10 shows the resulting gain over the BASE. We observe a different trend here compared to Fig. 8. Here the best performing strategy is MEDA and increasing the percentage of non-toxic samples (MEDA-90) for which the model receives the control variable $C_{\texttt{nont}}$ does not improve the average gain. This is because we have identified the optimal value of PRMNONT for MEDA and its

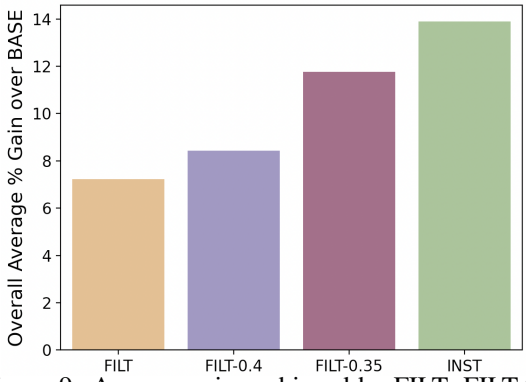

Figure 9: Average gains achieved by FILT, FILT-0.4, FILT-0.35 and INST over BASE across the eleven tasks on 357m-96m model configuration. We see that the average performance of the model improves when we filter more toxic samples and replace them with non-toxic samples. We illustrate that INST strategy still performs better than variations of filter strategy.

variations. We have not yet explored the optimal value of PRMNONT for INST.

We also experimented with using the raw toxicity scores from Perspective API directly without binning them as in case of MEDA for 357m-96m configuration. Specifically, in case of MEDA all the samples within a threshold get the same value, either 0.1 or 0.5. In this ablation (MEDA-R), the samples within a threshold would get the raw scores like 0.01 or 0.67 up to two decimal points. We observe that MEDA-R does not reduce as much toxicity compared to MEDA (toxicity probability: only 7% reduction by MEDA-R compared to 36% by MEDA; xpected maximum toxicity: 5% reduction by MEDA-R as opposed to 22% by MEDA).

5 Related Work

Finetuning-based Methods Pretrained LMs can be further finetuned using different training algorithms like domain-adaptive training methods (Gehman et al., 2020; Gururangan et al., 2020; Solaiman and Dennison, 2021; Wang et al., 2022) and reinforcement learning (Ouyang et al., 2022; Perez et al., 2022) on non-toxic data. These methods can only be employed after LMs are pretrained. These methods typically incur further costs of finetuning additional LMs (Krause et al., 2021; Liu et al., 2021), generating large amount of non-toxic data (Wang et al., 2022), or procuring human feedback (Ouyang et al., 2022). Our work on the other hand is targeted towards reducing toxicity by augmenting the pretraining corpus and hence will not incur additional cost after the LM is trained.

Decoding Time Algorithms They reduce toxicity of the generations at decoding time by altering the probabilities of certain tokens. Gehman et al. (2020) show a study on using PPLM (Dathathri et al., 2020), word-filtering, and vocabulary shifting (Keskar et al., 2019). Schick et al. (2021) use the internal knowledge of the LM to reduces the probability of generating toxic text. The GeDi approach (Krause et al., 2021) guides the generation at each step by computing classification probabilities for all possible next tokens. Liu et al. (2021) propose DEXPERTS which controls the generation with an"expert" LM trained on non-toxic data and "anti-expert" LM trained on toxic data. These techniques are efficient at reducing toxicity but fail to consider the underlying semantic meaning of the generated text at the sequence level. They may also reduce the utility of the LM at performing downstream tasks (Wang et al., 2022).

Analysis of Toxicity in Pretraining Data Large body of work analyzes the pretraining data and advocates for choosing it carefully (Gehman et al., 2020; Welbl et al., 2021; Bender et al., 2021). Gehman et al. (2020) provide an analysis of toxicity on a subset of pretraining data at a document-level. Our analysis (§2.1) of the entire pretraining corpus is at a sample-level and in agreement with Gehman et al. (2020). An analysis by Sap et al. (2019) reports that filtering data based on PerspectiveAPI could lead to a decrease in text by African American authors. Our proposed approaches (MEDA and INST) don't filter data. Additionally, Xu et al. (2021) present an analysis on different detoxification techniques like DAPT, PPLM, GeDi and filtering (Gururangan et al., 2020; Dathathri et al., 2020; Krause et al., 2021). They conclude that these techniques hurt equity and decrease the utility of LMs on language used by marginalized groups. These studies necessitate tackling toxicity at the pretraining data stage without filtering.

Ngo et al. (2021) present experiments by filtering toxic documents based on the loglikelihood of the text. Our work augments pretraining data with toxicity information.

6 Comparison with Prior Work

Prior work and this study uses different model configurations in terms of model parameters, pretraining data, number of samples, and hyperparameters. We show comparison with closest model configuration with our work. We only compare the relative

changes because the baselines are different for our work and Wang et al. (2022); Liu et al. (2021); Welbl et al. (2021). Also, PerspectiveAPI (PerspectiveAPI, 2022) update their models regularly and hence scores returned by it may change over time.

Finetuning-based Methods Wang et al. (2022) develop SGEAT which first generates large amount of non-toxic data using a pretrained LM (Smith et al., 2022) and then uses domain adaptive finetuning. They have the same model parameters and pretraining data as our models. We follow the same toxicity evaluation setup and similar setup for benchmark NLP tasks. We compare 357m-150m INST model (Table 2) with the SGEAT 357m model in Tables 1 and 3 of Wang et al. (2022). We see that toxicity probability is relatively reduced by a massive 60% for INST and 38% for SGEAT; accuracy for benchmark NLP tasks show a relative improvement of 0.9% for INST whereas SGEAT decreases the NLP utility by 1.4%; perplexity is relatively increased by 0.85% for INST and 9% for SGEAT. We would like to note that prior work (Gururangan et al., 2020; Wang et al., 2022; Ouyang et al., 2022) has not evaluated bias detection tasks and text generation task (E2E).

Decoding Time Algorithms Due to similar model configuration with Wang et al. (2022), we compare DExperts (reported in Table 2) with results on INST 1.3b-150m configuration (Table 3). We observe that toxicity probability gets relatively reduced by 69.5% for DExperts and 63.5% for INST; but accuracy of benchmark NLP tasks is significantly decreased by DExperts (15%) and only 1.3% by INST. Hence, even if decoding time algorithms provide a higher decrease in toxicity, they are not usable for general NLP tasks.

Filtering Methods Prior work (Welbl et al., 2021; Ngo et al., 2021) removes entire documents with toxicity above a threshold from the training set. FILT strategy replaces the toxic samples with equivalent number of non-toxic samples. To have a fair comparison, we train two models: (1) a baseline 357m parameter model (BASE-Doc), and (2) we filter documents (2.5%) with toxicity score above 0.5 and train a model (FILT-Doc) on the remainder 97.5% of the documents. FILT-Doc reduces expected maximum toxicity by 4.2% and toxicity probability by 4.6% compared to BASE-Doc. This demonstrates that FILT-Doc provides lesser relative gains in toxicity reduction compared to FILT and INST (FILT gives 7.3% and INST provides 28.1% reduction in expected maximum toxicity; FILT shows 16% and INST displays 59.7% reduction in toxicity probability for 357m-150m in Table 2). This shows that sample-level FILT is $\sim 11\%$ better than document-level FILT-Doc on toxicity probability. We observe that FILT-Doc loses utility on benchmark NLP tasks by 1% and loses bias detection capabilities by 8% compared to BASE-Doc.

Based on the above comparisons, we conclude that INST strategy developed in this work demonstrates massive reduction in toxicity while preserving the utility of the LM on benchmark NLP tasks as well as bias detection tasks.

7 Conclusion and Future Work

We develop two new strategies to reduce toxicity using data augmentation - MEDA and INST. Through extensive experiments, we demonstrate that MEDA and INST reduce toxicity probability substantially (54% and 61% respectively) while not compromising on the utility of the LM on five benchmark NLP tasks and four bias detection tasks. We also show that adding control variables does not compromise performance on E2E task.

In this work, we show how toxicity can be reduced in LMs by augmenting the pretraining data with toxicity information. We believe that this idea can be extended to other dimensions of social biases and hate speech. Prior work shows that adding instructions during finetuning can help various NLP tasks and improve the LMs capabilities to generalize for instructions on unseen tasks (Wei et al., 2021; Ouyang et al., 2022). We postulate that these observations can be applied to adding instructions to the pretraining data which can make INST generalizable to reduce different types of biases.

The key idea of adding relevant information to the pretraining data via instructions can be applied more broadly and opens new directions for future work. Future work can focus on controlling the generation by adding general instructions to the pretraining data. Current work has applied MEDA and INST on binary view of toxicity i.e something is toxic or non-toxic. Hence, another interesting direction is to explore the degrees of toxicity and incorporate it with MEDA and INST strategies. Future work can also evaluate the generalizability and applicability of INST strategy on more text generation tasks.

Limitations

The current studies presented in this work rely on PerspetiveAPI (PerspectiveAPI, 2022). PerspectiveAPI scoring has been shown to be biased against marginalized communities (Gehman et al., 2020; Welbl et al., 2021; Xu et al., 2021). This can impact the strategies developed in this work. But we would like to note that MEDA and INST techniques can be used with any other classifier which provides toxicity scores. Another limitation of this work is that it requires a reliable classifier which provides effective score of toxicity. If the classifier provides with inaccurate toxicity scores then it would impact the performance of MEDA and INST. To apply the strategies discussed in this work, we have to label the whole pretraining dataset. This is true even for FILT strategy. Although not a limitation, this is an artifact of working on curating pretraining dataset. We would also like to point out that the control variables introduced in this work can be used for both generating non-toxic content as well as toxic content. If we append sample with $C_{\texttt{tox}}$ control variable instead of $C_{\texttt{nont}}$ then the LM would generate toxic data. We would like to assert that the intended use of this technique is to generate text that is not toxic.

References

Emily M. Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? . In *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, FAccT '21, page 610–623, New York, NY, USA. Association for Computing Machinery.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 7432–7439.

Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

CommonCrawl. 2022. Common crawl. https://commoncrawl.org/. (Accessed on 10/18/2022).

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2020. Plug and play language models: A simple approach to controlled text generation. In *International Conference on Learning Representations*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 3356–3369.

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360.

Dirk Hovy and Shrimai Prabhumoye. 2021. Five sources of bias in natural language processing. *Language and Linguistics Compass*, 15(8):e12432.

Nitish Shirish Keskar, Bryan McCann, Lav R Varshney, Caiming Xiong, and Richard Socher. 2019. Ctrl: A conditional transformer language model for controllable generation. *arXiv preprint arXiv:1909.05858*.

Ben Krause, Akhilesh Deepak Gotmare, Bryan McCann, Nitish Shirish Keskar, Shafiq Joty, Richard Socher, and Nazneen Fatema Rajani. 2021. Gedi: Generative discriminator guided sequence generation. In *Findings of the Association for Computational Linguistics: EMNLP 2021*, pages 4929–4952.

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.

Kris McGuffie and Alex Newhouse. 2020. The radicalization risks of gpt-3 and advanced neural language models. *arXiv preprint arXiv:2009.06807*.

Helen Ngo, Cooper Raterink, João GM Araújo, Ivan Zhang, Carol Chen, Adrien Morisot, and Nicholas Frosst. 2021. Mitigating harm in language models with conditional-likelihood filtration. *arXiv preprint arXiv:2108.07790*.

Yixin Nie, Adina Williams, Emily Dinan, Mohit Bansal, Jason Weston, and Douwe Kiela. 2020. Adversarial nli: A new benchmark for natural language understanding. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4885–4901.

Jekaterina Novikova, Ondřej Dušek, and Verena Rieser. 2017. The E2E dataset: New challenges for end-to-end generation. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pages 201–206, Saarbrücken, Germany. Association for Computational Linguistics.

Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *arXiv preprint arXiv:2203.02155*.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Ngoc-Quan Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. 2016. The lambada dataset: Word prediction requiring a broad discourse context. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1525–1534.

Ethan Perez, Saffron Huang, Francis Song, Trevor Cai, Roman Ring, John Aslanides, Amelia Glaese, Nat McAleese, and Geoffrey Irving. 2022. Red teaming language models with language models. *arXiv preprint arXiv:2202.03286*.

PerspectiveAPI. 2022. Perspective | developers. https://developers.perspectiveapi.com/s/. (Accessed on 10/18/2022).

Shrimai Prabhumoye, Brendon Boldt, Ruslan Salakhutdinov, and Alan W Black. 2021a. Case study: Deontological ethics in NLP. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3784–3798, Online. Association for Computational Linguistics.

Shrimai Prabhumoye, Rafal Kocielnik, Mohammad Shoeybi, Anima Anandkumar, and Bryan Catanzaro. 2021b. Few-shot instruction prompts for pretrained language models to detect social biases. *arXiv preprint arXiv:2112.07868*.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2020. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740.

Maarten Sap, Dallas Card, Saadia Gabriel, Yejin Choi, and Noah A. Smith. 2019. The risk of racial bias in hate speech detection. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1668–1678, Florence, Italy. Association for Computational Linguistics.

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.

Timo Schick, Sahana Udupa, and Hinrich Schütze. 2021. Self-diagnosis and self-debiasing: A proposal for reducing corpus-based bias in nlp. *Transactions of the Association for Computational Linguistics*, 9:1408–1424.

Scikit-learn. 2022. Roc-auc-score. https://scikit-learn.org/stable/modules/generated/sklearn.metrics.roc_auc_score.html. (Accessed on 04/13/2022).

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *arXiv preprint arXiv:1909.08053*.

Shaden Smith, Mostofa Patwary, Brandon Norick, Patrick LeGresley, Samyam Rajbhandari, Jared Casper, Zhun Liu, Shrimai Prabhumoye, George Zerveas, Vijay Korthikanti, et al. 2022. Using deepspeed and megatron to train megatron-turing nlg 530b, a large-scale generative language model. *arXiv preprint arXiv:2201.11990*.

Irene Solaiman and Christy Dennison. 2021. Process for adapting language models to society (palms) with values-targeted datasets. *Advances in Neural Information Processing Systems*, 34:5861–5873.

Trieu H Trinh and Quoc V Le. 2018. A simple method for commonsense reasoning. *arXiv preprint arXiv:1806.02847*.

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

Boxin Wang, Wei Ping, Chaowei Xiao, Peng Xu, Mostofa Patwary, Mohammad Shoeybi, Bo Li, Anima Anandkumar, and Bryan Catanzaro. 2022. Exploring the limits of domain-adaptive training for detoxifying large-scale language models. *arXiv preprint arXiv:2202.04173*.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. 2021. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Johannes Welbl, Amelia Glaese, Jonathan Uesato, Sumanth Dathathri, John Mellor, Lisa Anne Hendricks, Kirsty Anderson, Pushmeet Kohli, Ben Coppin, and Po-Sen Huang. 2021. Challenges in detoxifying language models. In *Findings of the Associ-*

ation for Computational Linguistics: EMNLP 2021, pages 2447–2469.

Albert Xu, Eshaan Pathak, Eric Wallace, Suchin Gururangan, Maarten Sap, and Dan Klein. 2021. Detoxifying language models risks marginalizing minority voices. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2390–2397.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019a. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019b. Defending against neural fake news. *Advances in neural information processing systems*, 32.

A Analysis of PerspectiveAPI scores

Doc. Id	#chars	Doc. Score	2k chars	5k chars
1	3198	0.0816	0.1052	0.0816
2	7053	0.0778	0.0996	0.0827
3	4337	0.0806	0.0731	0.0806
4	3575	0.1293	0.1275	0.1293
5	2168	0.0763	0.0767	0.0763
6	9820	0.1395	0.1801	0.2051
7	9917	0.0851	0.2052	0.2428
8	3971	0.2400	0.2612	0.2400
9	9644	0.2586	0.3880	0.2843
10	6964	0.2208	0.3644	0.3546

Table 1: PerspectiveAPI scores of documents using three different ways. #chars denotes the number of characters in a document. Doc. Score is the PerspectiveAPI toxicity score when the entire document is passed. 2*k chars* displays the average PerspectiveAPI toxicity score when the document is split into chunks of 2k chars and 5*k chars* displays the average PerspectiveAPI toxicity score when the document is split into chunks of 5k chars. Doc. Id denotes the id of the document.

PerspectiveAPI accepts maximum text size per request of 20 KB. This is approximately 20k characters. We select 10 documents with less than 10k characters for the purpose of our analysis. This analysis aims to study the difference between PerspectiveAPI toxicity scores when we pass the whole document vs chunking the document and then averaging the scores for each chunk. We obtain PerspectiveAPI toxicity score in three ways: (1) we pass the whole document and get the PerspectiveAPI toxicity score (denoted as "Doc. Score" in Table 1), (2) we split the document into chunks of 2000 characters and then take the weighted average of PerspectiveAPI toxicty scores for all the chunks (denoted as "2k chars" in Table 1), and (3) we split the document into chunks of 5000 characters and then take the weighted average of PerspectiveAPI toxicity scores for all the chunks (denoted as "5k chars" in Table 1).

Table 1 shows the result of this analysis. We observe that all the three types of scores are different. More importantly the ranking between the documents changes if we consider each of the three approaches. For example if we rank the document ids from lowest score to highest toxicity score then the ranking according to the approaches are: (1) *Doc. Score* is $5, 2, 3, 1, 7, 4, 6, 10, 8, 9$ (2) 2*k chars* is $3, 5, 2, 1, 4, 6, 7, 8, 10, 9$ and (3) 5*k chars* is $5, 3, 1, 2, 4, 6, 8, 7, 9, 10$. This study shows that document longer than 20k characters cannot be split into multiple chunks to obtain an average Per-

Model	96m-samples					150m-samples				
	EMT	TP	NLP	BD	E2E	EMT	TP	NLP	BD	E2E
BASE	0.44	0.36	47.5	50.6	27.6	0.43	0.35	48.2	50.0	30.8
FILT	0.40	0.29	48.0	51.2	27.4	0.40	0.30	48.5	52.2	30.0
	↓8.1%	↓18.5%	↑1.1%	↑1.2%	↓0.8%	↓7.3%	↓16.0%	↑0.6%	↑4.4%	↓2.8%
MEDA	0.41	0.31	48.1	50.1	28.5	0.41	0.31	48.2	49.5	30.7
	↓5.9%	↓13.2%	↑1.4%	↓1.0%	↑3.2%	↓4.8%	↓11.0%	↑0.0%	↓1.0%	↓0.7%
INST	0.42	0.33	47.9	50.2	28.9	0.42	0.33	48.7	51.1	29.7
	↓2.8%	↓6.8%	↑0.8%	↓0.9%	↑4.9%	↓1.9%	↓5.4%	↑0.9%	↑2.3%	↓3.7%
Experiment using control variable $C_{\texttt{nont}}$										
MEDA	0.33	0.18	-	-	28.3	0.33	0.17	-	-	30.7
	↓24.0%	↓49.8%			↑2.6%	↓23.9%	↓51.2%			↓0.5%
INST	0.31	0.15	-	-	29.8	0.31	0.14	-	-	29.7
	↓29.0%	↓59.3%			↑7.8%	↓28.1%	↓59.7%			↓3.5%

Table 2: Results for 357m parameter models on all the metrics. EMT is Expected Maximum Toxicity; TP is Toxicity Probability; NLP indicates the average of accuracy on five benchmark NLP tasks; BD displays the average AUC on four bias detection tasks; and E2E shows the Rouge-L scores of the LMs on the E2E task. For benchmark NLP tasks, bias detection tasks and E2E task we show the relative percentage improvement over BASE with a ↑% and decrement with a ↓%. For the expected maximum toxicity and toxicity probability, we show the improvement with ↓% because lower is better for these metrics. We may observe that two strategies obtain the exact same score but there is a difference in their relative percentages. This is because these scores are computed up to 4 decimal digits but we only report scores up to 2 decimals here.

spectiveAPI score.

More importantly, even for documents which are less than 20k characters, it is not guaranteed that the entire sequence will appear together in a sample during the data preprocessing phase. Hence, first obtaining PerspectiveAPI score and then splitting the documents into samples of sequence length 2000 tokens would yield inaccurate toxicity scores for the samples. Hence, our approach is focused on sample-level toxicity scoring for providing the LM with precise toxicity information. This impacts our MEDA and INST strategies which rely on guiding the LM at sample-level about toxicity information.

B Details of Main Results

Table 2 and 3 show the results for the eleven tasks with and with the control variable $C_{\texttt{nont}}$ for 357m and 1.3b parameter models respectively. EMT is Expected Maximum Toxicity; TP is Toxicity Probability; NLP indicates the average of accuracy on five benchmark NLP tasks; BD displays the average AUC score on four bias detection tasks; and E2E shows the Rouge-L scores of the LMs on the E2E task. For benchmark NLP tasks, bias detection tasks and E2E task we show the relative percentage improvement over BASE with a ↑% and decrement with a ↓%. For the expected maximum toxicity and toxicity probability, we show the improvement with ↓% because lower is better for these metrics. In Tables 2 and 3, we may observe that two strategies obtain the exact same score but there is a difference in their relative percentages. This is because these scores are computed up to 4 decimal digits but we only report scores up to 2 decimals here.

We calculate the relative percentage difference compared to BASE for all the twelve metrics across the eleven tasks - expected maximum toxicity, toxicity probability, accuracy of five NLP tasks, AUC scores of four bias detection tasks, and Rouge-L for E2E task. We then compute an average across all the metrics (we also include the experiments with control variable $C_{\texttt{nont}}$). These aggregated results are shown in Fig. 4. Fig. 4 shows the average percentage gains achieved by each strategy across the eleven tasks.

C Hyper-parameter Details

All the LMs trained in this work are GPT-style (Brown et al., 2020) Transformer architectures (Vaswani et al., 2017) trained with Megatron toolkit (Shoeybi et al., 2019). We use BPE tokenization with a vocabulary of size 50256. All the models are trained on sequence length of 2048 tokens. Note that our samples are of size 2000 tokens. We leave 48 tokens for adding either raw scores for MEDA or instructions for INST. Note that the baseline is also trained on same samples of 2000 tokens. We pad the extra spaces with a PAD_TOKEN.

Model	96m-samples					150m-samples				
	EMT	TP	NLP	BD	E2E	EMT	TP	NLP	BD	E2E
BASE	0.44	0.37	52.6	53.0	30.7	0.44	0.37	54.4	53.8	31.1
FILT	0.40	0.30	52.9	55.9	29.9	0.41	0.31	54.5	53.2	31.9
	↓8.5%	↓18.8%	↑0.6%	↑5.5%	↓2.5%	↓7.4%	↓16.5%	↑0.2%	↓1.1%	↑2.6%
MEDA	0.42	0.33	53.0	57.2	31.8	0.42	0.34	53.9	53.5	33.0
	↓4.7%	↓10.7%	↑0.8%	↑7.9%	↑3.7%	↓4.2%	↓9.1%	↓0.9%	↓0.6%	↑6.1%
INST	0.43	0.34	53.3	53.9	30.6	0.42	0.34	53.7	54.9	31.7
	↓3.6%	↓9.7%	↑1.3%	↑1.7%	↓0.2%	↓3.7%	↓9.2%	↓1.3%	↑2.0%	↑2.1%
Experiment using control variable $C_{\texttt{nont}}$										
MEDA	0.32	0.16	-	-	31.9	0.32	0.16	-	-	33.6
	↓27.5%	↓58.1%			↑4.2%	↓26.8%	↓57.4%			↑8.2%
INST	0.31	0.15	-	-	31.3	0.31	0.14	-	-	32.6
	↓30.2%	↓62.7%			↑2.1%	↓30.0%	↓63.5%			↑4.9%

Table 3: Results for 1.3b parameter models. EMT is Expected Maximum Toxicity; TP is Toxicity Probability; NLP indicates the average of accuracy on five benchmark NLP tasks; BD displays the average AUC on four bias detection tasks; and E2E shows the Rouge-L scores of the LMs on the E2E task. For benchmark NLP tasks, bias detection tasks and E2E task we show the relative percentage improvement over BASE with a ↑% and decrement with a ↓%. For the expected maximum toxicity and toxicity probability, we show the improvement with ↓% because lower is better for these metrics. We may observe that two strategies obtain the exact same score but there is a difference in their relative percentages. This is because these scores are computed up to 4 decimal digits but we only report scores up to 2 decimals here.

357m parameter models We train them with 24 layers, with a hidden size of 1024 and 16 attention heads. We use $\texttt{max} - \texttt{position} - \texttt{embeddings}$ of 2048; 162761 warmup samples; a learning rate of $3.0e^{-4}$ with minimum learning rate of $3.0e^{-5}$. We use cosine learning decay style. Additionally, we use clip-grad = 1.0, weight-decay = 0.1, adam-beta1 = 0.9, and adam-beta2 = 0.95. Each of these models are trained on 64 A100 GPUs with 40GB memory. The models with 96m samples are trained for 54 GPU hours and models with 150m samples are trained for 84 GPU hours.

1.3b parameter models We train them with 24 layers, with a hidden size of 2048 and 32 attention heads. We use max-position-embeddings = 2048 with 244141 warmup samples; a learning rate of $2.0e^{-4}$ with a minumum learning rate of $2.0e^{-5}$ and cosine decay style. We use clip-grad = 1.0, weight-decay = 0.1, adam-beta1 = 0.9, and adam-beta2 = 0.95. Each of these models are trained on 64 A100 GPUs with 40GB memory. The models with 96m samples are trained for 113 GPU hours and models with 150m samples are trained for 176 GPU hours.

Bounds for the new variables We describe the bounds for LOWTHRESH, HIGHTHRESH, PRMTOX and PRMNONT variables introduced in this work.

$$0 < \text{LOWTHRESH} < 1$$
$$0 < \text{HIGHTHRESH} < 1$$
$$0 \leq \text{PRMTOX} \leq 1$$
$$0 \leq \text{PRMNONT} \leq 1$$

Note that PRMTOX = 0 means that no samples above the HIGHTHRESH are augmented with $C_{\texttt{tox}}$; and PRMTOX = 1 means that all the samples above the HIGHTHRESH are augmented with $C_{\texttt{tox}}$. Similarly, PRMNONT = 0 implies that no samples below LOWTHRESH are modified; and PRMNONT = 1 means that all the samples below LOWTHRESH are augmented with $C_{\texttt{nont}}$.

Number of Shots for Tasks Table 4 shows the number of shots used as context for each task following the setups in Brown et al. (2020); Smith et al. (2022); Prabhumoye et al. (2021b).

Task	# of Shots
LAMBADA (Paperno et al., 2016)	15
ANLI (Nie et al., 2020)	50
Winogrande (Sakaguchi et al., 2020)	50
PiQA (Bisk et al., 2020)	50
Hellaswag (Zellers et al., 2019a)	20
Bias Detection (Prabhumoye et al., 2021b)	32

Table 4: Number of shots used as context for each task.

Model	150m-samples			
	EMT	TP	NLP	BD
BASE	0.43	0.35	64.9	54.7
FILT	0.39	0.28	64.8	61.0
	↓8.0%	↓19.6%	↓0.1%	↑11.5%
INST	0.41	0.30	65.3	61.7
	↓4.5%	↓13.4%	↑0.7%	↑12.7%
Experiment using control variable C_{nont}				
INST	0.28	0.11	-	-
	↓34.0%	↓69.5%		

Table 5: Results for 8.3b parameter models trained with 150 million samples.

Model	96m-samples				
	EMT	TP	NLP	BD	E2E
BASE	0.44	0.36	47.5	50.6	27.6
FILT	0.40	0.29	48.0	51.2	27.4
	↓8.1%	↓18.5%	↑1.1%	↑1.2%	↓0.8%
FILT-0.4	0.38	0.25	47.4	50.0	28.8
	↓13.1%	↓30.8%	↓0.3%	↓1.1%	↑4.3%
FILT-0.35	0.37	0.23	48.0	50.4	28.4
	↓15.1%	↓36.8%	↑1.1%	↓0.4%	↑2.9%
INST	0.42	0.33	47.9	50.2	28.9
	↓2.8%	↓6.8%	↑0.8%	↓0.9%	↑4.9%
Experiment using control variable C_{nont}					
INST	0.31	0.15	-	-	29.8
	↓29.0%	↓59.3%			↑7.8%

Table 6: Results for 357m-96m configuration on all the metrics for variations of FILT such as FILT-0.4 and FILT-0.35 in comparison with INST.

D Ablation Experiments

Scaling the Model Size Table 5 shows results for 8.3 billion parameter models which use BASE, FILT and INST strategies on 357m-150m model configuration.

FILT Variations Table 6 shows the results for BASE, FILT, FILT-0.4, FILT-0.35 and INST for all the 357m-96m model configuration on all eleven tasks. These results are aggregated and presented in Fig. 9.

MEDA Variations Table 7 shows the results for BASE, MEDA-11, MEDA, MEDA-90 and INST for 357m-96m model configuration on all eleven tasks. These results are aggregated and presented in Fig. 10.

INST Variations Table 8 shows the results for BASE, INST-11, INST-50 and INST for all the four model configuration on all eleven tasks. These results are aggregated and presented in Fig. 8.

Model	96m-samples				
	EMT	TP	NLP	BD	E2E
BASE	0.44	0.36	47.5	50.6	27.6
MEDA-11	0.42	0.33	47.1	52.6	28.4
	↓4.6%	↓8.8%	↓0.8%	↑4.1%	↑3.0%
MEDA	0.41	0.31	48.1	50.1	28.5
	↓5.9%	↓13.2%	↑1.4%	↓1.0%	↑3.2%
MEDA-90	0.42	0.33	47.0	47.6	28.6
	↓2.9%	↓8.2%	↓1.1%	↓6%	↑3.8%
INST	0.42	0.33	47.9	50.2	28.9
	↓2.8%	↓6.8%	↑0.8%	↓0.9%	↑4.9%
Experiment using control variable C_{nont}					
MEDA-11	0.35	0.20	-	-	28.4
	↓20.7%	↓43.5%			↑3.0%
MEDA	0.33	0.18	-	-	28.3
	↓24.0%	↓49.8%			↑2.6%
MEDA-90	0.31	0.14	-	-	28.4
	↓28.6%	↓59.9%			↑3.0%
INST	0.31	0.15	-	-	29.8
	↓29.0%	↓59.3%			↑7.8%

Table 7: Results for 357m-96m configuration on all the metrics for MEDA and INST in comparison with MEDA-11 and MEDA-90.

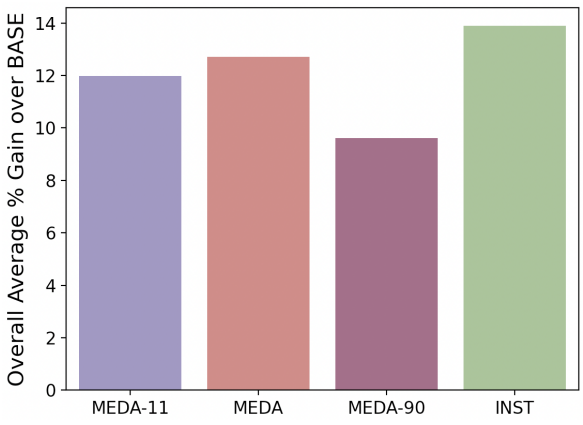

Figure 10: Average the gains achieved by MEDA-11, MEDA, MEDA-90, and INST over BASE across the eleven tasks for 357m-96m model configuration.

Model	96m-samples					150m-samples				
	EMT	TP	NLP	BD	E2E	EMT	TP	NLP	BD	E2E
Experiments with 357m parameter models										
BASE	0.44	0.36	47.5	50.6	27.6	0.43	0.35	48.2	50.0	30.8
INST-11	0.41	0.32	46.6	50.9	28.1	0.42	0.34	48.7	50.7	28.7
	↓5.9%	↓11.8%	↓1.8%	↑0.5%	↑1.9%	↓2.0%	↓3.9%	↑1.0%	↑1.4%	↓6.8%
INST-50	0.41	0.32	47.9	49.9	28.2	0.42	0.33	48.3	49.1	29.5
	↓5.6%	↓11.9%	↑0.9%	↓1.5%	↑2.3%	↓3.0%	↓6.8%	↑0.2%	↓1.8%	↓4.3%
INST	0.42	0.33	47.9	50.2	28.9	0.42	0.33	48.7	51.1	29.7
	↓2.8%	↓6.8%	↑0.8%	↓0.9%	↑4.9%	↓1.9%	↓5.4%	↑0.9%	↑2.3%	↓3.7%
Experiment using control variable $C_{\texttt{nont}}$ for 357m parameter models										
INST-11	0.36	0.23	-	-	28.3	0.38	0.25	-	-	29.0
	↓18.5%	↓36.8%			↑2.7%	↓13.0%	↓27.4%			↓6.0%
INST-50	0.32	0.16	-	-	28.0	0.34	0.18	-	-	29.5
	↓26.7%	↓54.6%			↑1.5%	↓22.3%	↓48.2%			↓4.4%
INST	0.31	0.15	-	-	29.8	0.31	0.14	-	-	29.7
	↓29.0%	↓59.3%			↑7.8%	↓28.1%	↓59.7%			↓3.5%
Experiments with 1.3b parameter model										
BASE	0.44	0.37	52.6	53.0	30.7	0.44	0.37	54.4	53.8	31.1
INST-11	0.41	0.32	52.9	54.1	33.5	0.42	0.33	54.3	54.0	31.0
	↓6.3%	↓13.3%	↑0.5%	↑2.2%	↑9.1%	↓4.9%	↓11.0%	↓0.1%	↑0.3%	↓0.3%
INST-50	0.41	0.32	53.4	54.5	30.8	0.42	0.34	53.9	54.6	32.6
	↓6.3%	↓14.3%	↑1.4%	↑2.9%	↑0.4%	↓3.7%	↓8.4%	↓0.8%	↑1.4%	↑4.9%
INST	0.43	0.34	53.3	53.9	30.6	0.42	0.34	53.7	54.9	31.7
	↓3.6%	↓9.7%	↑1.3%	↑1.7%	↓0.2%	↓3.7%	↓9.2%	↓1.3%	↑2.0%	↑2.1%
Experiment using control variable $C_{\texttt{nont}}$ for 1.3b parameter models										
INST-11	0.32	0.15	-	-	34.2	0.33	0.18	-	-	32.1
	↓28.3%	↓59.6%			↑11.7%	↓24.3%	↓51.4%			↑3.3%
INST-50	0.31	0.14	-	-	30.9	0.32	0.15	-	-	32.9
	↓29.6%	↓61.4%			↑0.7%	↓27.3%	↓58.4%			↑6.0%
INST	0.31	0.15	-	-	31.3	0.31	0.14	-	-	32.6
	↓30.2%	↓62.7%			↑2.1%	↓30.0%	↓63.5%			↑4.9%

Table 8: Results for 357m and 1.3b parameter models on all the metrics for INST and its variations INST-11 and INST-50. For benchmark NLP tasks, bias detection tasks and E2E task we show the relative percentage improvement over BASE with a ↑% and decrement with a ↓%. For the expected maximum toxicity and toxicity probability, we show the improvement with ↓% because lower is better for these metrics.