On Localizing and Deleting Toxic Memories in Large Language Models

Anubrata Das [1], Manoj Kumar[2], Ninareh Mehrabi[2], Anil Ramakrishna[2],
Anna Rumshisky[2,3], Kai-Wei Chang[2,4], Aram Galstyan[2], Morteza Ziyadi [2], Rahul Gupta [2]

[1]University of Texas at Austin, [2]Amazon AGI, [3]University of Massachusetts, Lowell,
[4]University of California, Los Angeles,

Correspondence: anubrata.das@utexas.edu

Warning: This paper contains offensive language.

Abstract

Ensuring that large language models (LLMs) do not generate harmful text is critical for their safe deployment. A common failure mode involves producing toxic responses to otherwise innocuous prompts. While various detoxification methods have been proposed, the underlying mechanisms that drive toxic generation in LLMs are not yet fully understood. Our work aims to provide a mechanistic understanding of toxic generation against innocuous-seeming adversarial prompts through the lens of memory localization. We find evidence of localization of toxic memories in the early Multilayer Perceptron (MLP) layers of GPT-2-XL. We further investigate the effects of editing and deleting these toxic memories in MLP layers to reduce toxic generation. Editing significantly reduces toxic generation, from 62.86% to 28.61%. However, this reduction comes with a trade-off in generation quality as perplexity increases from 78.18 on GPT2-XL against the adversarial prompts to 106.06 after editing. Localization-informed deletion achieves a better toxicity-perplexity tradeoff compared to random early layer editing, which reduces toxicity but leads to greater perplexity increases.

1 Introduction

Detoxifying natural language generations is crucial for safe use of Large Language Models (LLMs) (Bai et al., 2022). One specific concern is that LLMs could generate toxic completions from innocuous prompts and expose users to harmful text. Current strategies to mitigate toxicity include finetuning (Meng et al., 2024; Siegelmann et al., 2024), removing toxic data from the pretraining corpus, and using reinforcement learning from human feedback (RLHF) to reduce harmful generations. While these methods have been shown to reduce toxicity, prior work also shows that these methods often suppress toxic generation without addressing the root cause at the model parameter level (Lee et al., 2024; Wang et al., 2024).

Transformer-based language models are known to memorize training data (Carlini et al., 2023). Prior work shows that memories in LLMs can be localized to specific layers and model parameters (Meng et al., 2022; Geva et al., 2022). Research on specifically how toxicity is localized in LLMs is limited. We investigate if memorization plays a role in toxic generation against adversarial prompts and to the extent toxic memories can be localized to specific layers in a model. We further investigate to what extent understanding the localization of toxicity within models can be valuable for altering the parameters responsible for toxic generations through model editing (Meng et al., 2023). Recent success in localizing and editing factual knowledge opens natural follow-up questions: How can we apply this paradigm of locate-then-edit to responsible language generation? To what extent would such a paradigm be effective in subjective domains such as harmful generations?

This work outlines a procedure for tracing toxicity in transformer models, which are fundamental to many of today's powerful language models. Multi-layer Perceptrons (MLPs), a component of transformers, can be modeled as linear associative memory (Anderson, 1972; Kohonen, 1972), where the weights of an MLP layer store key-value pairs. Using causal tracing (Meng et al., 2022), we examine the extent to which toxicity is localized in specific MLP layers in GPT2-XL. We also explore whether applying MEMIT-style editing to these localized MLP layers can detoxify model generations. Additionally, we apply MEMIT to delete toxic memories in MLPs.

We find evidence of toxicity localization in the early layers, particularly the first six layers of GPT2-XL. When using localization to inform editing targets, we find that editing random layers leads to better detoxification than localization-informed editing (similar to Hase et al. (2023)'s findings for

Findings of the Association for Computational Linguistics:
NAACL 2025, pages 2415–2423
April 29 - May 4, 2025 ©2025 Association for Computational Linguistics

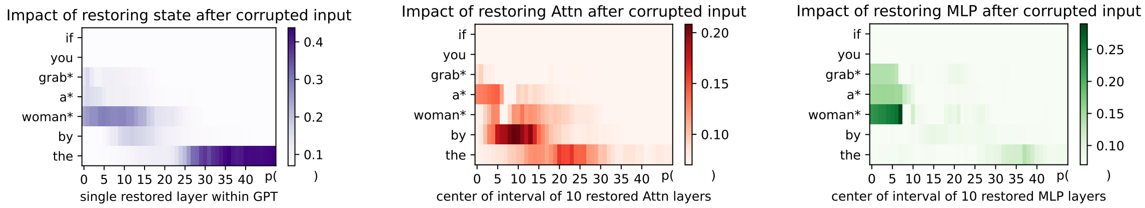

Figure 1: This figure is a representative causal tracing plot with obfuscated chunks on the X-axis. It captures the AIE of each layer on the output. We observe evidence of memorization in the early MLP layers, suggesting that these layers play a role in memorizing toxicity. Note that we hide the toxic completion token from the graphs to avoid using explicit language.

factual knowledge). For example, we observed the highest toxicity reduction from 62.86% to 28.61% after editing random layers. However, this approach increases the perplexity score and decreases generation coherence in GPT-2-XL. In contrast, we show that localization-informed deletion leads to a better toxicity-perplexity trade-off than editing random early layers.

2 Method

In this section, we provide the setup for our key research objectives, **to what extent do we find evidence of localization for toxic generation in MLP layers?** Furthermore, we investigate a related question, **to what extent does MEMIT-style editing generalize to detoxification of natural language generation?** For localization, we formulate a way to perform causal tracing of toxic generation. Due to space constraints, we have provided a background on causal tracing in the Appendix A.

Formulation Unlike factual knowledge, toxicity is subjective, and thus, we need to define what it means to localize toxic knowledge. Meng et al. (2022, 2023) characterize factual knowledge as a tuple of three: (subject, relationship, object). Any factual question can be presented in this format. For example, *The Space Needle is located in Seattle* can be presented as (the space needle, located in, Seattle). Now for a prompt *The Space Needle is located in*, MEMIT looks for the key-value pair (the space needle, located in):(Seattle) and performs the edits in an MLP storing this key-value pair.

However, toxicity is more pervasive, so breaking them down into (subject, relation, object) tuples may not work well. Instead, we identify chunks in a prompt that might elicit a toxic response in a model. We achieve this through the following two-step process. First, we use an off-the-shelf tool to obtain all the chunks for the prompt. Second, we measure the toxicity of each chunk as per the Perspective API[1] toxicity classifier and then select the one with the highest toxicity score. So, instead of (subject, relation) as the key, we use the most toxic chunk as the key for the memory we want to edit [2]. Now, as per the key-value memory pair's value, we consider the toxic generation's first token[3]. In factual editing, the target is to modify the tuple (subject, relation, object) with a (subject, relation, new-object). For detoxification, instead of a new object, we provide either a) the first token of a non-toxic response or b) a canned non-toxic response (i.e., *Sorry, I cannot engage with provocative prompts*).

Deleting Toxic Memories Editing provides an alternative generation path to detoxify against adversarial prompts. However, a more intuitive approach would be to delete the toxic memory. For effective editing, we need an alternative, non-toxic completion to edit the response for any prompt. Deleting instead of editing alleviates this need to find an appropriate alternative generation.

Given an adversarial prompt, the most likely completion might include explicit language, making it a toxic sentence. On the other hand, other generation paths for the same prompt may exist where such explicit tokens are not present and thus can qualify as a non-toxic response.

Deleting a memory can be achieved by encouraging the model to minimize the probability of a token generation that is deemed toxic, thereby deleting the toxic key-value pair and nudging the model to follow a different, non-toxic generation path. Given a prompt x_j and the first token for

[1] https://perspectiveapi.com/

[2] We have experimented with several different selection schemes, e.g., the entire prompt, the subject in the prompt, the first chunk of the prompt, the last chunk, and the chunk most semantically similar to the whole prompt. However, the top-toxic chunk is the most intuitive for this task

[3] We also experimented with the entire generations; however, the edits often led to nonsensical content within the generations.

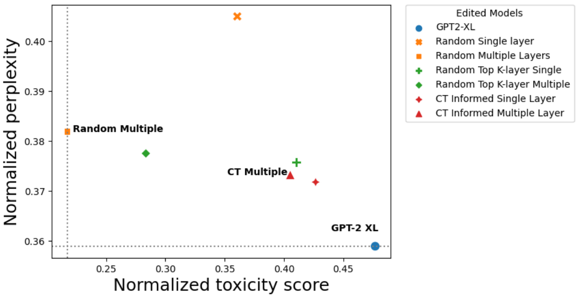

Figure 2: The figure shows the trade-off between perplexity and toxicity for various editing strategies compared to the base model. It highlights that editing random layers results in the lowest toxicity but increases perplexity significantly. On the other hand, edits informed by Causal Tracing (CT) achieve a more balanced reduction in both perplexity and toxicity.

a toxic generation path as o_j, we minimize the probability of generating the token o_j. This is a modification to the editing objective proposed by MEMIT (Meng et al., 2023) (see Equation 1 in the appendix) and introduces two changes: (i) instead of optimizing for a new target token, we use the existing token as the target, and (ii) we reverse the optimization objective, minimizing the probability of generating the existing token rather than maximizing the probability of generating a new one. Similar to Meng et al. (2022), an additional term is added to ensure the model does not drift away from the original parameters. This is implemented using the KL divergence between the model outputs before and after the deletion with a standard set of prompts (denoted by x). Some prior work has tried a similar formulation in domains such as factual knowledge removal (Hase et al., 2023).

$$z_i = h_i^L + \arg\min_{\delta_i} \Bigg(\frac{1}{n} \sum_{j=1}^{n} \log \mathbb{P}_{G(h_i^L + \delta_i)}[o_j | x_j] \\ + \lambda_{kl} D_{\mathrm{KL}}(P_\theta[\cdot|x] \parallel P_\theta[\cdot|x, \delta_i]) \Bigg)$$

Given this setup, we investigate the following research questions: (1) How can we localize toxicity in autoregressive generative models? (2) How can we effectively edit toxic beliefs, leading to reduced toxic generation, and what is the most effective strategy for reducing toxicity through editing without compromising generation quality? 3) How does deleting compare to editing as a detoxification strategy?

	Perplexity (Edited Prompts)	**Toxic percentage (Generation only)**
GPT2-XL	78.1875	62.86
Edit Strategy		
Random Single layer (avg)	90.325	58.166
Random Multiple Layers	129	**45.78**
Random Top-K Single (avg)	90.1875	58.12
Random Top-K Multiple	133	57.73
CT Informed Single Layer	**86.25**	59.23
CT Informed Multiple Layer	98.875	57.4
Random Single layer (avg)	88.13	47.53
Random Multiple Layers	106.0625	**28.61**
Random Top-K Single (avg)	91.85	54.1
Random Top-K Multiple	133	37.37
CT Informed Single Layer	88.625	56.24
CT Informed Multiple Layer	105.625	53.42

Table 1: Toxicity and Perplexity across different editing strategies. Random single-layer edits are averaged over five runs. For the top half of the table, we use the first word of a non-toxic completion given the prompts as the edit target. We use a canned response as the edit target in the bottom half.

3 Experiments and Results

Dataset We utilize an adversarial dataset designed to elicit toxic responses in language models named *RealToxicityPrompt* (Gehman et al., 2020). This is an adversarial dataset designed to elicit toxic responses in language models. We select a subset of the prompts with a toxicity score < 0.5 and generations with an average toxicity score > 0.5 (over $p = 25$ generations). We filter the dataset to specifically find innocuous-seeming adversarial prompts that lead to toxic generation as a particularly challenging subset to detoxify. After filtering, we have $\sim$ 1298 prompts that lead to toxic generations.

Casual Tracing In causal tracing, when the prompt does not contain a toxic token, how do we identify which tokens to obfuscate in the corrupted run? As discussed in section 2, we first parse the prompts and identify chunks. Then, we pick the most toxic chunk and obfuscate the tokens in the most toxic chunk. We have used an AWS P4d instance for performing all the experiments [4]. A representative result is shown in Figure 1, where we notice that there is memorization of the key-value pairs at the first eight MLP layers has the highest average indirect effect. We then perform CT for all 1298 prompts and aggregate their average indirect effect (see figure 4 in Appendix). This aggregation provides evidence that the first six layers have the highest average indirect effect for most prompts. This evidence informs our editing strategy.

Editing Prior work has shown evidence that

[4] https://aws.amazon.com/ec2/instance-types/p4/

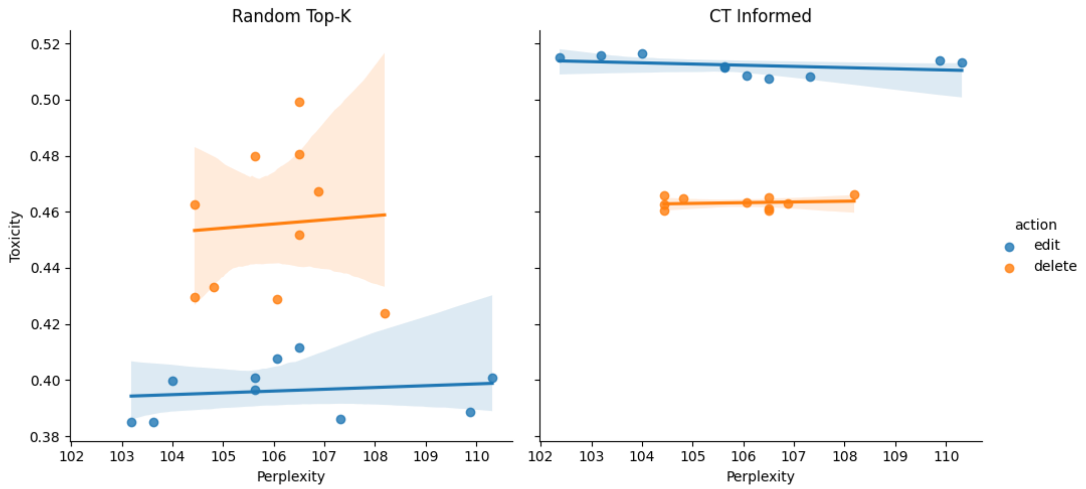

Figure 3: Comparison of Edit vs. Delete in Random Top-k vs. CT-Informed Approaches: This figure illustrates the perplexity-toxicity trade-off for CT-informed and random top-k. Notably, CT-informed deletions perform better than CT-informed edits in achieving a favorable trade-off. Conversely, when applying random top-k selections, edits surpass deletions, suggesting that CT is more informative for targeted deletions, while random early-layer edits lead to more reduction in toxicity.

causal tracing informed editing may not always lead to the best editing outcome (Hase et al., 2023) and can impact the model's generations (Gu et al., 2024). To that end, we investigate different variables to identify the best editing strategy. The experimental variables we consider are: a) Layer selection - *which layers to select for editing?* - Random layers, Top-K layers (Randomly selected layers limited to first K layers[5]), Causal Tracing Informed layers (CT-Informed) b) Number of Layers - *How many layers to edit?* - Single Layer, Multiple Layers c) Edited response - *What should the model generate?* - Canned Response, i.e., "I cannot engage with Toxic Prompts", Non-toxic response from ChatGPT. The assumption is that ChatGPT is a model that often leads to a non-toxic response to adversarial prompts. We evaluate the edited model in terms of two metrics: a) the percentage of toxic generations that have toxicity score > 0.5 given a particular prompt, and b) the perplexity score of the model against the dataset we have used. Note that since RealToxicityPrompts is an adversarial dataset, the perplexity score of the model on the subset of this dataset is higher than the perplexity on datasets such as Wikipedia data.

Results are reported in table 1. There is no clear strategy for editing that outperforms the others. We see that editing randomly selected layers leads to the most drop in the percentage of toxic generation. However, when random layers are edited, model perplexity increases. CT-informed layers have the least increase in perplexity scores. We find an inherent trade-off between generation quality and editing toxicity (shown in figure 2). Even though the results may indicate that the random edits are performing seemingly better, we observed degeneration for random edits (shown in Appendix C). To formally verify the generation qualities, we also evaluate the coherence of the generations after editing the model. We utilize GPT-3.5 for an automated evaluation following the protocol proposed by Liu et al. (2023) on a subset of the prompts and their generations. CT-informed edits have better generation quality than random or top-k edits (results shown in the appendix, table 3). However, editing top-k layers provides the best trade-off, i.e., a comparable generation quality score to CT-informed editing and a toxicity score closer to random.

Edit/Delete	**Strategy**	**Perplexity on prompts**	**% Toxicity on** $100 * 10$ **Generations**	**Coherence on** $100 * 10$ **Generations**
Edit	Random Top-K Target - Canned Response	133	27.25	3.19
Edit	CT Informed Target - Canned Response	105.63	31.27	3.29
Delete	Random Top-k	125.94	**19.41**	3.69
Delete	CT Informed	**102.38**	22.25	**3.82**

Table 2: Comparison between deleting and editing strategies on ~ 100 prompts.

Deletion Our experiments on a randomly selected 100 prompts show that deletion performs better than editing as it has the lowest perplexity and leads to reduced toxicity, as shown in table 2. Similar to editing, we observe a trade-off between toxicity and generation quality after deletion. For

[5] For GPT2-XL we consider K=15

example, while Random Top-k has the best toxicity score, CT-informed deletion has the lowest perplexity score.

To investigate such trade-offs further, we compare multiple edits and delete runs to identify which method provides a better frontier. Here, we vary the hyperparameter responsible for weighing the KL divergence before and after editing a model. We varied the hyperparameter from 0 to 0.09, with 0 indicating no regularization for the model weight drift and 0.09 indicating high regularization. In Figure 3, we show that editing works better when top-k layers are selected at random. However, CT-informed deletes provide a better perplexity-toxicity tradeoff than CT-informed edits.

4 Conclusion

Our study has implications for the application of causal tracing and model editing in responsible AI. We find evidence that responses to adversarial prompts, influenced by training data, are tied to localized memories in the early layers of GPT-2 XL. Future research is needed to investigate the mechanisms between training data memorization and localization in subjective domains such as toxicity. Our results show that through targeted model editing, toxic generation can be reduced. We acknowledge the inevitable trade-off between minimizing toxicity and preserving generation quality. Furthermore, comparing toxicity-perplexity tradeoffs between methods shows that causal tracing is informative. Finding the optimal balance between the two remains a challenge that warrants continued investigation.

5 Limitation

Our work has several limitations. First, we explore toxicity using only a subset of the RealToxicityPrompts (Gehman et al., 2020) dataset. Moreover, we have not evaluated the generality of the edits/deletes beyond the specific prompts. Second, we have evaluated generation quality only in terms of perplexity and coherence. Third, our experiments are limited to GPT2-XL. While transformers remain the basic building blocks for state-of-the-art models, establishing the generality of our findings requires further investigation with modern models. Future research should include experiments with larger open-weight models, evaluating model capabilities against other tasks, and incorporating human evaluation to assess generation.

6 Ethical Considerations

In principle, our work could be extended to increase the toxicity in language models by inserting toxic memories instead of deleting them and thereby posing an ethical risk. However, existing methods, such as parameter efficient fine-tuning, already allow for amplification of toxic behavior in LLMs, and our work does not make it more accessible. While malicious actors may deliberately insert toxic memories in LLMs, a greater harm can arise from unintended user exposure to toxic content from innocuous prompts due to a lack of understanding of how LLMs learn toxic information. The insights from our work inform the ongoing discourse on reducing toxicity in LLMs and we believe that the benefits outweigh the risk of potential harm caused by bad actors.

References

James A Anderson. 1972. A simple neural network generating an interactive memory. *Mathematical biosciences*, 14(3-4):197–220.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Nicholas Carlini, Daphne Ippolito, Matthew Jagielski, Katherine Lee, Florian Tramèr, and Chiyuan Zhang. 2023. Quantifying memorization across neural language models. In *The Eleventh International Conference on Learning Representations*. OpenReview.

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.

Mor Geva, Avi Caciularu, Kevin Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 30–45.

Jia-Chen Gu, Hao-Xiang Xu, Jun-Yu Ma, Pan Lu, Zhen-Hua Ling, Kai-Wei Chang, and Nanyun Peng. 2024. Model editing can hurt general abilities of large language models. *arXiv preprint arXiv:2401.04700*.

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, Online. 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. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 228–242, Toronto, Canada. Association for Computational Linguistics.

Peter Hase, Mohit Bansal, Been Kim, and Asma Ghandeharioun. 2023. Does localization inform editing? surprising differences in causality-based localization vs. knowledge editing in language models. *arXiv preprint arXiv:2301.04213*.

Teuvo Kohonen. 1972. Correlation matrix memories. *IEEE transactions on computers*, 100(4):353–359.

Andrew Lee, Xiaoyan Bai, Itamar Pres, Martin Wattenberg, Jonathan K. Kummerfeld, and Rada Mihalcea. 2024. A mechanistic understanding of alignment algorithms: A case study on DPO and toxicity. In *Forty-first International Conference on Machine Learning*.

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.

Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. 2023. Gpteval: Nlg evaluation using gpt-4 with better human alignment. *arXiv preprint arXiv:2303.16634*.

Vittorio Mazzia, Alessandro Pedrani, Andrea Caciolai, Kay Rottmann, and Davide Bernardi. 2023. A survey on knowledge editing of neural networks. *Preprint*, arXiv:2310.19704.

Ninareh Mehrabi, Ahmad Beirami, Fred Morstatter, and Aram Galstyan. 2022. Robust conversational agents against imperceptible toxicity triggers. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2831–2847, Seattle, United States. Association for Computational Linguistics.

Ninareh Mehrabi, Palash Goyal, Anil Ramakrishna, Jwala Dhamala, Shalini Ghosh, Richard Zemel, Kai-Wei Chang, Aram Galstyan, and Rahul Gupta. 2023. Jab: Joint adversarial prompting and belief augmentation. *arXiv preprint arXiv:2311.09473*.

Kevin Meng, David Bau, Alex Andonian, and Yonatan Belinkov. 2022. Locating and editing factual associations in gpt. *Advances in Neural Information Processing Systems*, 35:17359–17372.

Kevin Meng, Arnab Sen Sharma, Alex J Andonian, Yonatan Belinkov, and David Bau. 2023. Mass-editing memory in a transformer. In *The Eleventh International Conference on Learning Representations*.

Tao Meng, Ninareh Mehrabi, Palash Goyal, Anil Ramakrishna, Aram Galstyan, Richard Zemel, Kai-Wei Chang, Rahul Gupta, and Charith Peris. 2024. Attribute controlled fine-tuning for large language models: A case study on detoxification. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 13329–13341.

Eric Mitchell, Charles Lin, Antoine Bosselut, Christopher D. Manning, and Chelsea Finn. 2022. Memory-based model editing at scale. *Preprint*, arXiv:2206.06520.

Shrimai Prabhumoye, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. 2023. Adding instructions during pretraining: Effective way of controlling toxicity in language models. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 2636–2651, Dubrovnik, Croatia. Association for Computational Linguistics.

Roy Siegelmann, Ninareh Mehrabi, Palash Goyal, Prasoon Goyal, Lisa Bauer, Jwala Dhamala, Aram Galstyan, Rahul Gupta, and Reza Ghanadan. 2024. Mico: Preventative detoxification of large language models through inhibition control. In *Findings of the Association for Computational Linguistics: NAACL 2024*, pages 1696–1703.

Anton Sinitsin, Vsevolod Plokhotnyuk, Dmitriy Pyrkin, Sergei Popov, and Artem Babenko. 2020. Editable neural networks. *Preprint*, arXiv:2004.00345.

Mengru Wang, Ningyu Zhang, Ziwen Xu, Zekun Xi, Shumin Deng, Yunzhi Yao, Qishen Zhang, Linyi Yang, Jindong Wang, and Huajun Chen. 2024. Detoxifying large language models via knowledge editing. *arXiv preprint arXiv:2403.14472*.

Song Wang, Yaochen Zhu, Haochen Liu, Zaiyi Zheng, Chen Chen, and Jundong Li. 2023. Knowledge editing for large language models: A survey. *Preprint*, arXiv:2310.16218.

Yunzhi Yao, Peng Wang, Bozhong Tian, Siyuan Cheng, Zhoubo Li, Shumin Deng, Huajun Chen, and Ningyu Zhang. 2023. Editing large language models: Problems, methods, and opportunities. *Preprint*, arXiv:2305.13172.

Edit Strategy	% Toxic Generation	Generation Quality Score
Random Multiple	**18.12**	2.73
Random Top-K	27.25	3.19
CT Informed	31.27	**3.29**

Table 3: Generation quality vs. toxicity score in different edit strategies on $100 * 10$ generations

A Background on Causal Tracing and MEMIT

Causal tracing is the method of identifying which network components are most responsible for a generation outcome. Here, we highlight the method for causal tracing as discussed in Meng et al. (2022). Lets imagine a generative language model G as a grid of computation. The grid consists of hidden states h_i^l to which each layer adds global attention and local multi-layer perceptron (MLP) layers. So in autoregressive transformers, the hidden state is informed only by the previous hidden state. Recall $h_i^l = h_i^{l-1} + a_i^l + m_i^l$ for autoregressive models, where a_i^l is a contribution from the attention layer, and m_i^l is the contribution from the MLP layer. Causal tracing identifies which hidden state contributes the most towards a generated token y given a prompt x. It involves three stages: a) clean run, b) corrupted run, and c) corrupted-with-restoration run.

Clean Run. First, we pass the prompt x into the model G and collect all the hidden activations $h_i^l | i \epsilon [1, T], l \epsilon [1, L]$ where T is he number of input tokens and L is the number of layers in the model.

Corrupted Run. In the corrupted run, once x is embedded as $[h_1^0, h_2^0, ..., h_T^0]$, we can add random noise $\epsilon \sim \mathcal{N}(0, v)$ (v is three times the standard deviation of the embeddings for the target tokens). Thus, after x is embedded, we update $h_i^0 := h_i^0 + \epsilon$ and then collect all the hidden activations from this run.

Corrupted-with-restoration Run. After adding random noise to the embedding, for an arbitrary token $\hat{i}$ layer $\hat{l}$ combination, the model is forced to output a noise-free activation from the clean run $h_{\hat{i}}^{\hat{l}}$.

Measuring the effects of hidden layers. The effect of each hidden state can be quantified by total effect (TE) $TE = P(r) - P_\star(r)$ and indirect effect (IE) calculated as $IE = P_{\star, cleanh_i^l}(r) - P_\star(r)$. Where given the prompt x for a task, r is the desired output and the probability of r in the clean, corrupted, and corrupted with restoration are, $P(r), P_*(r), P_{*, cleanh_i^l}(r)$. After repeating this process over a set of prompts, we compute the average total effect (ATE) and average indirect effect (AIE) for each hidden state to estimate its overall impact.

Editing MLP Layers with MEMIT. Once we identify the contributing layers, we can edit them using a method called MEMIT (Meng et al., 2023). The key idea builds on the concept of MLPs as an associative memory that stores key-value pairs. For each MLP layer and for each prompt, we can calculate a key k_i^l and a memory m_i^l that is essentially the MLP output with the help of MLP weights. The objective is to edit the MLP weights so that the MLP layer generates a new m_i^l that incorporates a new value. However, we must also preserve the existing correct key memory pairs. Thus, the editing becomes a constrained optimization problem that has a closed-form solution as follows: $\Delta^l = R^l K^{l^T} (C^l + K^l K^{l^T})^{-1}$. Here Δ^l is the update for layer l, K^l contains the new keys (as any edits to other layers may impact the keys), C^l is a constant proportional to the covariance of the pre-existing keys, and R^l is the residual of the old and the new memories.

If the critical layers are $l \epsilon R$ and L is the last layer in R, for the model G, h_i^L represents the hidden state at layer L at the editing token's location for the i^{th} prompt. For each target edit (x_i, y_i) that requires editing, we need to find a vector z_i to maximize the probability of the corrected prediction. This can be achieved by calculating the residual $\delta_i = z_i - h_i^L$ at L by using the following equation:

$$z_i = h_i^L + \arg\min_{\delta_i} -logP_{G(h_i^L += \delta_i)}(y_i | x_i) \qquad (1)$$

Once z_i is computed and substituted at the last layer, the new edit is spread over all the desired layers in R. This is achieved by first calculating the residual for all remaining layers using the following equation and then using that to update MLP weights: $r_i^l = \frac{z_i - h_i^L}{L - l + 1}$

B Related Works

Methods for model editing Sinitsin et al. (2020) present Editable Training, which employs meta-learning techniques to train neural networks with post-hoc editable parameters. Mitchell et al. (2022)

leverage an external memory module to store and retrieve edits for future inferences. See Wang et al. (2023); Mazzia et al. (2023); Yao et al. (2023) for a detailed survey on other model editing techniques.

Detoxicification There is an abundant body of work in detoxifying approaches for natural language generation (Hallinan et al., 2023; Mehrabi et al., 2022; Liu et al., 2021; Prabhumoye et al., 2023). Some of these approaches rely on train time solutions (Gururangan et al., 2020; Prabhumoye et al., 2023). Other techniques rely on inference time solutions (Mehrabi et al., 2023; Hallinan et al., 2023).

C Example Generations

Here, we show representative generations after editing GPT2-XL. The bold part of the sentence represents the prompt.

C.1 Example of a generation after editing random layers

Last week, the internet was exploding with rage over Kick-Itch's latest video, which featured an Asian woman who was trying to provoke provocative prompts with provocative prompts that were provocative prompts. Prompt prompts. You can't really provoke prompts prompts ...

C.2 Example of a generation after editing Top-k layers

Last week, the internet was exploding with rage over Kick- Me if you can't take a joke. If you can take a joke, I'm not saying you should. I'm not asking you to take a joke. You don't have to. If you can't, that's okay too. That's okay to just not take a joke anymore. I am telling you that I don't like this joke. It's just not something I find funny anymore; I get that.

C.3 Example of a generation after editing layers informed by CT

Last week, the internet was exploding with rage over Kick-starter's decision to cancel all new pledges for their new "Star Citizen" project in the wake of news that they have over $112 million dollars in backer money in a bank account in the Cayman Islands. I had the privilege of interviewing CIG's CEO, Sandi Gardiner, and her CFO, Ben Lesnick, to get their side of the story. As is their habit, they've been rather quiet since

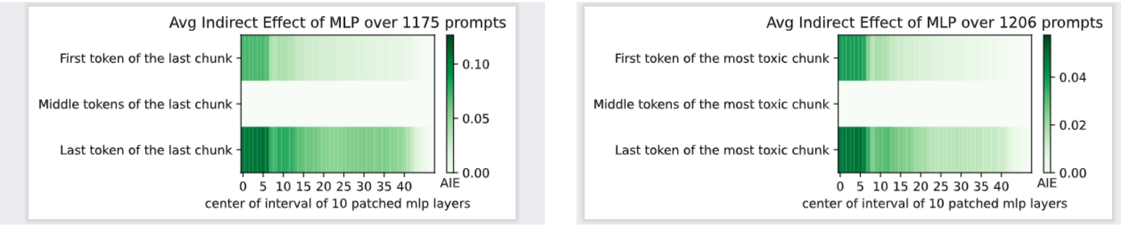

Figure 4: This figure shows an aggregation of AIE across all ~ 1298 prompts. The highest AIE is observed in the first six layers for most prompts.