"How Much Can You Trust Your Friend, Who Knows Everything? Analyzing the Vulnerabilities of Large"

Hasan Mustafa
IIT Delhi / Hauz Khas
aiy247541@scai.iitd.ac.in

Taki Hasan
IIT Delhi / Hauz Khas
aib242295@scai.iitd.ac.in

Abstract

We investigate toxic text generation by LLMs. We train a single layer toxic vector classifier on Jigsaw Toxic Comment Classification Dataset. We use this classifier as our probing model to identify the value vectors which encode toxicity information in each layer of the transformer. We finetune 2 different models namely Counter Speech Model and Toxic Model(both on top of GPT-2) and further train a Counter Speech Generation Model from scratch(without pretraining) to enable comparison in extreme case scenario. We the carry out analysis on the space of the identified toxic value vectors. Our analysis shows interesting properties of these vectors, we comment on the nature and span of these vectors, and investigate how they behave when exposed to prompts that elicit toxic responses.

1 Introduction

Large Language models are powerful tools that excel at various natural language processing tasks. Yet the working of such transformer based models are poorly understood(as is the case with most neural models).

A paricularly concerning aspect of this black box nature of large LLMs is that it can be used to generate harmful content by the user, or it can unintentionally give harmful or offensive response.

In this work(as in prior works) we look at the mechanisms of the model which dictate such behavior when exposed to harmful content. LINK TO GITHUB

2 A look at Transformer Architecture

A decoder only transformer(such as GPT) has two major components, the attention layer and the MLP layer. The attention layer applies a linear transform to the inputs, while the MLP layer stores bulk of the knowledge, by learning non linear relationships (Geva et al., 2022). Architectures like GPT use two layer Neural Networks as the MLP layer. The first layer projects the input to a higher dimensional input space and the second layer projects it back to the original dimension of the residual stream. The resulting output is added to the residual stream.

Thus, one can view each transformer layer as feeding to a residual stream and in effect updating the distribution over tokens.

$$\tilde{x} = \tilde{x} + MLP(o) \tag{1}$$

Where $\tilde{x}$ is the residual stream and o is the output from the multi-head self attention block.

As in (Lee et al., 2024) we view the final MLP layer of each transformer layer being "value vectors", which are weighted by the inputs to the layer.

$$[W_1, \cdots, W_{4096}] \begin{bmatrix} o_1 \\ o_2 \\ \vdots \\ o_{4096} \end{bmatrix} = \sum_{i=1}^{4096} o_i W_i \tag{2}$$

Where W_i are 1024 dimensional vectors spanning the column space of the second MLP weight matrix $W : 4096 \rightarrow 1024$. Since a linear combination of these value vectors updates the residual stream, it is reasonable to hypothesise that these weight matrix column vectors encode the represtations that might lead to toxicity/ other behaviour patterns of the LLM.

3 Hypothesis 1

3.1 Toxicity Probe

We train a linear probe on Jigsaw Toxic comment classification dataset and achieve an accuracy of 94% with F1 score of 61%. The input data for training the probe was the final output of the model before applying the unembedding layer averaged across timesteps. A learning rate of 0.001 was used, along with 10 steps of gradient accumulation to enable the gradient flow from larger batches as our chosen batch size(16) could lead to no negative samples in some batches.

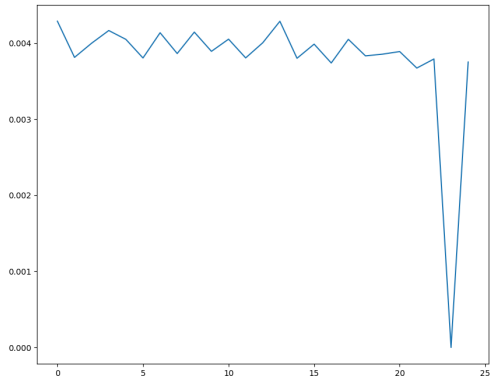

Figure 1: Mean Toxicity Score(of top 100 most toxic values in the entire model) for each layer.

Toxic Value (Index, Layer)	Vocabulary space projection top 5
770, 19	' sh*t', ' a*s', ' cr*p', ' f**k'
2192, 10	'ensive', 'angible', ' Eleven', 'multipl'
1544, 13	' godd', ' f**king', ' Niet', 'F**k'
4051, 7	'hips', 'age', 'loro', 'agg'

Table 1: Toxic Words Generate by projecting the value vectors identified by our probe onto the vocabulary space.

3.2 Nature of Toxic Vectors

We determine the nature of toxic vectors by projecting the most toxic vectors onto the vocabulary space. We find that these indeed correspond to toxic and offensive words. A sample of most toxic words is summarized in Table 1. We find(in agreement with previous works) that these values are spread across different layers of the model this is shown in Figure 1. We surprisingly find that none of the toxic values are present in the 23^{rd} layer.

The probe trained in (Lee et al., 2024) is slightly better than ours so we use their probe in further experiment. We might refer to our probe and will expicityl mention when we do so.

3.3 Span of toxic Vector

To study the span of the toxic vector we compute the singular value decomposition of the value vectors. We find that the matrix is a full rank matrix, with signular values decaying almost linearly. We perform projection of the eigenvectors on to the vocabulary spcae and see the effect of removing top-k

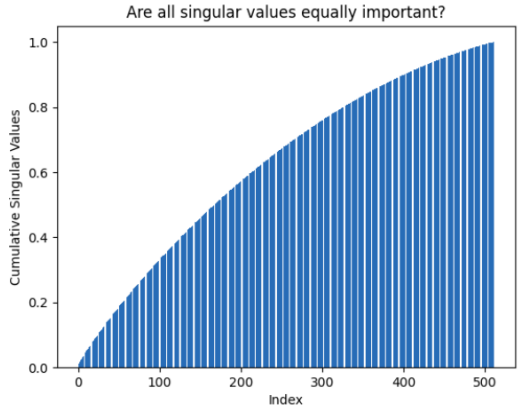

Figure 2: We plot the cumulative sum of the singular values of the 512 most toxic values. We find that the cumulative sum grows rather smoothly, and has a slight concave shape(indicating that singular vectors become slightly less important). We thus can not identity any prominent vectors to analyze in this subspace.

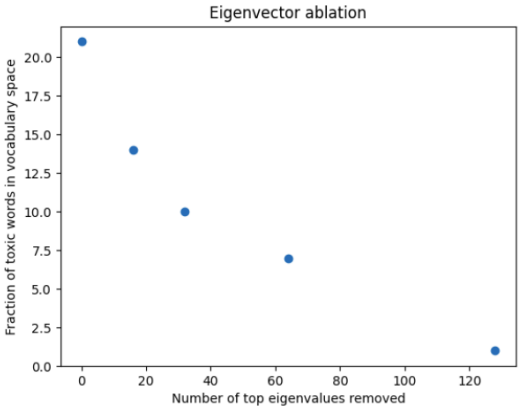

Figure 3: Fraction of toxic words generated by next top-10 eigenvalues after removing the k most toxic values.

singular values on the vocabulary space projections outputs. We further manually annotate the top-5 vocabulary tokens for each of the ablation cases and present the result as fraction of toxic tokens in Figure 3.

3.4 Counter Speech Generation Model

We train a counter speech generation model following the methodology in (onl). We utilize the CONAN multitarget dataset, which contains hate speech of various categories including DISABLED, JEWS, LGBT+, MIGRANTS, MUSLIMS, PEOPLE OF COLOR (POC), WOMEN. We train the model for 3 epochs with Adam optimizer. We create the dataset by joining the sentences <HATE SPEECH> <EOS> <COUNTER SPEECH> and training the model on auto-regressive language

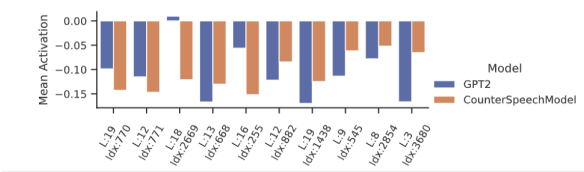

Figure 4: Activation Drop for Counter Speech Model

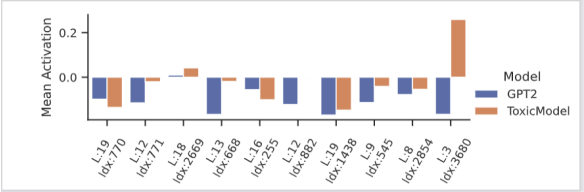

Figure 5: Activation Drop for Toxic Model

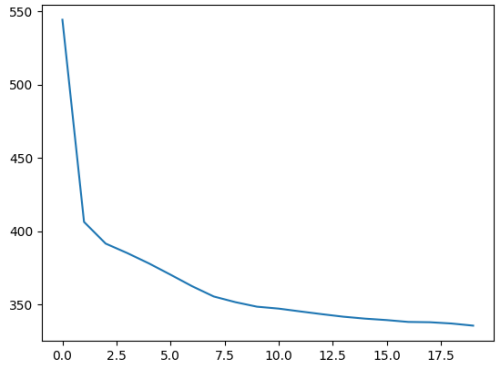

Figure 6: Counter Speech Model Validation Loss

modeling loss.

3.5 Consistency of Toxic Vectors

We find that even after finetuning the model parameters are almost same. However how these value vectors are utlizied by the model shifts significantly. To illustrate the difference, we train a second model(referred to as the Toxic Model) apart from the counter speech generation model, on only hate speech data. We then check how the default gpt-2 model compares with these two models in terms of the activation of these toxic value vectors.

Toxic model clearly activates the toxic value vectors more compared to base GPT model while the counter speech generation model does not activate the value vectors as much as base GPT.

4 Hypothesis 2

4.1 Training Models

We train a counter speech generation model from scratch(Referred to as Scratch Model) as well as finetune the gpt-2 model on counterspeech dataset. The loss vs epochs curve is shown in Figure 6 for the model trained from scratch.

4.2 Measuring activation drop

To measure activation drop we use a dataset that elicits toxic responses from GPT-2(following the method in (Lee et al., 2024)). The measured quantity is the average activation over many samples.

$$mean\_activation = \frac{1}{N}\sum_{i}^{N}\sigma(x_l^i W_k) \quad (3)$$

Some excerpts from the dataset are shown below. These prompts are known to cause the model to generate toxic token.

> "Boyega would later respond to those racist nobodies on his Instagram"
>
> "A racist radio ad featuring an Asian man named 'Ping Pong' calling"

By measuring the mean activation, we can estimate whether that value vector is being promoted or supressed by the model on an average.

4.3 Are all value vectors involved in producing non-toxic outputs?

Our analysis finds that while all value vectors are present in the model, both before and after finetuning, their activation changes. We find(as in Figure 4 and Figure 5) that after fine tuning on couter speech generation, the activations become negative i.e. they supress the generation of toxic values. We can contrast this with the opposite scenario where we finetune on only toixc data, we find that activations are positive and greater than the previous model.

Thus we conclude that key vectors play a significant role in generation.

Acknowledgments

Work was largely done together by the authors with each author working on all parts of the problem. The major contributions are as follows:

- Taki Hasan: Fine Tuning Models, SVD analysis, Hypothesis 2.
- Hasan Mustafa: Training Model from scratch and training probe, data preprocessing, Hypothesis 1.

Code was adapted from following sources

- https://github.com/ajyl/dpo_toxic
- https://nnsight.net/notebooks/tutorials/logit_lens/
- Training GPT-2
- Documentation of HuggingFace and TransformerLens

References

Contextual counterspeech generation.

Mor Geva, Avi Caciularu, Kevin Ro Wang, and Yoav Goldberg. 2022. Transformer feed-forward layers build predictions by promoting concepts in the vocabulary space. *Preprint*, arXiv:2203.14680.

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. *Preprint*, arXiv:2401.01967.