SENTENCE LEVEL EMBEDDING DETOXIFICATION VIA TOXIC COMPONENT REMOVAL

Andrew Jian Wang

Charlottesville, Virginia

Bachelor of Science, University of Virginia, 2021

A Thesis submitted to the Graduate Faculty

of the University of Virginia in Candidacy for the Degree of

Master of Science

Department of Computer Science

University of Virginia

May 2022

Yangfeng Ji, Chair

Haifeng Xu

Jundong Li

Sentence Level Embedding Detoxification via Toxic Component Removal

Andrew Jian Wang

(ABSTRACT)

When state of the art pre-trained language models are trained on internet discourse, they may unintentionally encode and perpetuate some of the hateful and toxic behavior known to exist within the training data. We hypothesize that the encoding of toxicity can be explained by a "toxic subspace" within sentence-level language model embeddings, the existence of which suggests that toxic features follow some underlying pattern and are thus removable. To identify this toxic subspace, we propose a method to find patterns in toxic lexical markers. Through our experiments, we demonstrate that the subspace we find can separate toxic and non-toxic examples in the latent space. We use this subspace to extract a toxic component within sentence embeddings, and demonstrate that the removal of this component causes the number of toxic representations to decrease. These results are promising and suggest that encoders can be reasonably detoxified.

Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

Transformer language models have recently found great success in tasks such as text classification, owing in large part to the large amounts of data used to train the model (Devlin et al. 2018; Liu et al. 2019; Wang et al. 2020; Song et al. 2020). Much of this data is derived from online sources without much vetting, and a major consequence is that training data contains toxic language (Gehman et al. 2020). As a result, language models learn toxic behavior (Wallace et al. 2019). Removing toxic text directly from the training data has proven difficult due to the amount of the training data present (Roller et al. 2020), therefore recent works consider post-hoc methods to remove toxic information from pre-trained language model output.

One post-hoc method method of note is the rephrasing of toxic sentences into non-toxic sentences. In style transfer, non-toxic rephrasing has been achieved via denoised auto-encoders (Laugier et al. 2021). The task of non-toxic rephrasing has also been framed as a modified paraphrasing task, yielding promising results (Dale et al. 2021). Both approaches make use of the encoder-decoder model architecture, where the encoder captures the context of a toxic sentence and feeds it to the first layer of the decoder to rephrase.

Since encoders capture the characteristics of a sentence, a potential way to facilitate non-toxic rephrasing is to directly detoxify the encoder of an encoder-decoder paraphraser. However, it remains unclear to what extent we can detoxify the encoder, or

even if it is practical. Thus, before we can hope to experiment with our envisioned paraphraser, we must *first* establish and ascertain that it is practical to detoxify the encoder.

In this work, we hypothesize that an encoder can be detoxified by removing the *toxic component* from encoder embeddings. Based on this hypothesis, we propose a novel yet simple way to detoxify language model encoders. Toxicity is often concentrated in lexical markers within a sentence. For instance, in the sentence

you're a complete idiot if that's what you think

the occurrence of the word "idiot" strongly suggests that this sentence is toxic. Instances of these lexical markers in a sentence follow some predictable pattern and can generalize to a subspace. To identify this subspace, we will use principal component analysis to extract toxic latent dimensions. We project toxic sentence embeddings onto this subspace to obtain the toxic component which can then be used to remove toxic information from the embedding.

We find that the subspace we derive using our approach is adept at explaining toxicity, to the point of being able to separate toxic and non-toxic examples in 2-D latent space.

To directly evaluate the effect of our approach on the encoder, we focus our analysis on sentence embeddings. Consequently, for sentence level embeddings, we avoid using word-level toxicity evaluation frameworks and word-level similarity metrics. Instead, we use sentence-level toxicity classification tasks to gauge performance. Suppose, for instance, that a toxicity classifier classifies a certain sentence as being toxic. After we apply our detoxification method, the classifier should no longer classify the sentence as toxic. We can think of this ideas as comparing the number of toxic predictions before and after toxic component removal. We observe that our method can successfully

reduce the number of toxic classifications.

We thus highlight two major contributions our paper makes.

1. A method to identify the toxic subspace
2. A method to use this toxic subspace to isolate and remove a toxic component from sentence-level embeddings.

Chapter 2

Review of Literature

The success of our toxic component removal is dependent on the quality of our subspace, which in turn is dependent on the quality of toxic lexical markers identified in each training example. To that end, we find that there are two main methods of identifying lexical methods. The first, more traditional method, uses a predefined list of "bad words" and searches for them in text. However, because this approach does not take into account context, it cannot guarantee that all toxic features will be identified, nor can it guarantee that the toxic features identified are toxic in the context they appear. Instead, we favor approaches that identify toxic lexical markers using context, as is often the case in more recent works on toxic sentence rephrasing (Laugier et al. 2021; Dale et al. 2021).

The task of removing an attribute component from embeddings using some underlying subspace has been well studied in the context of gender bias (Bolukbasi et al. 2016; Manzini et al. 2019; Ravfogel et al. 2020; Liang et al. 2020). The goal of debiasing is to prevent the encoder from capturing certain attributes of the text that may perpetuate bias. We can follow a similar framework to prevent the encoder from capturing lexical markers of toxicity. However, we note that gender debiasing and encoder detoxification operate under a different set of assumptions.

Unlike gender bias, where lexical markers for gender are ostensibly a finite set of gender pronouns, what constitutes a valid lexical marker for toxicity is nebulous at

best. For instance, how much a word affects toxicity often depends on the context it appears in. Further complicating the issue, encoders sometimes spuriously treat protected groups as lexical markers for toxicity (Dixon et al. 2018; Zhou et al. 2021). Any proposed method to detoxify an encoder must first account for these complications.

Chapter 3

Methodology

Our proposed method of text detoxification consists of three major components. First, we use a synthetic parallel corpus to identify a toxic subspace. Second, we project sentence representations (eg. pooled encoder hidden layer outputs) onto the toxic subspace to distill toxic information into a "toxic component." To create a detoxified representation we subtract the toxic component from the original representation.

3.1 Synthetic Parallel Corpus

To construct the toxic subspace, we require the preparation of a parallel toxic vs non-toxic training corpus. We construct this corpus by masking lexical markers in toxic text. Because of the difficulty in identifying lexical markers for toxicity, we propose to solve this problem by letting a toxicity classifier use the context of a sentence to identify and mask lexical markers within, instead of using a predefined list of markers. The strongest lexical markers in each toxic sentence are iteratively masked until the classifier identifies the sentence as non-toxic.

To identify the lexical markers for toxicity, we use an ensemble of two different toxicity classification models. The first model we use is the debiased toxicity classifier proposed by Dixon et al. (2018). Research in toxicity classification models has shed insight on how they can make biased decisions, such as high false-positive (toxic)

prediction rates for sentences that mention protected groups in a non-toxic way (cite papers). The unintended consequence of this phenomenon is that the very systems meant to protect protected groups may end up flagging their discourse as toxic and censoring them. To avoid this problem in our approach, we use the debiased model proposed by Dixon et al. (2018). The model is debiased by training on an adversarial training set, where the toxic examples mentioning protected groups are balanced with an equal of non-toxic examples mentioning protected groups.

However, in our experiments, we found that the debiased model achieves a precision of 0.9407 at the cost of having a recall of only 0.5080 on our toxicity test set, missing nearly 50% of toxic sentences. This implies one of two possibilities: either our data quality is poor, or the model is too conservative. Since we use reputable and well-cited datasets, we believe the latter to be the likelier of the two. Thus, we augment the identification of lexical markers with the fine-tuned RoBERTa-Large toxicity classified by Dale et al. (2021).

We use the debiased model to identify the most the toxic lexical marker. When feeding some toxic sentence through a toxicity classifier, the classifier outputs a probability distribution for each class. We can mask individual tokens in the sentence to investigate the effect a token has on this distribution. We define the most toxic lexical marker to be some token in the sentence that, when masked, causes the largest overall decrease in probability of belonging to the toxic class. Additionally, we can iterate on this process by masking one additional token each iteration. We provide an algorithm of our approach in 1 and several random examples of the synthetic corpus in Table 3.1. In Algorithm 1, MaskModel refers to the debiased model, and EvalModel refers to the Dale et al. 2021 RoBERTa classifier.

Algorithm 1 Synthetic Corpus Creation

Require: $\mathbf{S} = \{S_1 \dots S_n\}$ ▷ Let $\mathbf{S}$ represent a set of toxic sentence strings
Require: MaskModel, EvalModel
Ensure: $\{S_1 \dots S_n\}$ is non-toxic
  for $S_i \in \mathbf{S}$ **do**
    while $\Pr(\text{EvalModel}(S_i) = \text{toxic}) > 0.5$ **do**
      candidates, logits $\leftarrow \{\}$
      for $j \leftarrow 1$ to $|S_i|$ **do**
        $\text{candidates}_j \leftarrow \text{Mask}(S_i, j)$ ▷ Mask j-th word in S_i
        $\text{logits}_j \leftarrow \Pr(\text{MaskModel}(\text{candidates}_\text{j}) = \text{toxic})$
      end for
      $S_i \leftarrow \text{candidates}(\text{argmin}(\text{logits}))$
    end while
  end for

3.2 Subspace Identification and Removal

Intuitively, our goal is to learn a set of coordinate vectors that can explain toxicity within text. This goal has two main parts: 1) quantifying toxicity, and 2) using our definition of toxicity to identify coordinate vectors.

We first address the first part of the goal and outline a method of quantifying toxicity in text using our synthetic dataset. Suppose we feed every toxic example in the synthetic dataset through an encoder. Thus for a single example with *sequence-length* number of tokens, the final hidden layer yields an output with dimension *sequence-length* $\times$ *embedding-dim*. We can pool the output into a single dimension by averaging along all the individual token-level contextual embeddings, yielding a pooled embedding of dimension 1 $\times$ *embedding-dim*. We can repeat this procedure on the non-toxic sentences in our parallel synthetic dataset. For the sake of clarity we will refer to the set of n toxic pooled embeddings as $\mathbf{T} = \{T_1, T_2, ..., T_n\}$, where the dimension of of $\mathbf{T}$ is $n \times$ *embedding-dim*, and non-toxic pooled embeddings $\mathbf{N} = \{N_1, N_2, ..., N_n\}$, where the dimension of $\mathbf{N}$ is the same as the dimension of $\mathbf{T}$. Because

Toxic Sentence	**Synthetic Non-Toxic Sentence**
we can start by not letting these idiots into our country	we can start by not letting these [MASK] into our country
tell you what put whatever stupid stuff you want on your roof	tell you what put whatever [MASK] stuff you want on your roof
just make sure you are the dumb sot it [UNK] when it [UNK]	just make sure you are the [MASK] sot it [UNK] when it [UNK]
don't peddle your crap please	don't peddle your [MASK] please
trump has been and continues to be a disgrace	trump has been and continues to be a [MASK]

Table 3.1: Five random examples from the synthetic dataset.

our synthetic dataset is parallel, where each row is comprised of a toxic sentence and it's synthetically derived non-toxic sentence, we can say that T_1 corresponds to N_1, T_2 corresponds to N_2, so on and so forth.

We define toxicity locally, on a per example basis, as the vector difference between a toxic pooled embedding and its corresponding non-toxic pooled embedding, $T_i - N_i$, for the i-th row of the synthetic dataset. Let us call this difference vector d_i, where d_i also has dimension $1 \times$ *embedding-dim*. We can understand d_i to capture the information necessary to make non-toxic embedding N_i into toxic embedding T_i. More specifically, this difference captures both the direct effect of the toxic features as well as their impact on the surrounding context within the training example. To consider why this is the case, let us revisit the toxic pooled embedding T_i and its corresponding non-toxic pooled embedding N_i. We previously defined T_i and N_i to be the average of the individual token level contextual embeddings from the encoder's last hidden layer output. Suppose T_i is the mean of four token-level embeddings, $(t_1 + t_2 + t_3 + t_4)/4$, where t_1 through t_4 each represent a token level embedding. Suppose $N_i = (t'_1 + t'_2 + m + t'_4)/4$, where the third token of the original sentence was

masked and thus caused a change to the surrounding contextual embeddings. Let us rewrite the difference of T_i and N_i using these terms.

$$\begin{aligned} T_i - N_i &= \frac{(t_1 + t_2 + t_3 + t_4)}{4} - \frac{(t'_1 + t'_2 + m + t'_4)}{4} \\ &= \frac{(t_1 - t'_1) + (t_2 - t'_2) + (t_3 - m) + (t_4 - t'_4)}{4} \end{aligned}$$

We can see that the difference between T_i and N_i can be written in terms of $t_3 - m$, which quantifies the direct effect of masking the third token, and $t_1 - t'_1$, $t_2 - t'_2$, and $t_4 - t'_4$, which quantifies the secondary effects on the surrounding words caused by masking the third token.

Having defined toxicity as the difference between T_i and N_i, we can now derive a set of coordinate vectors. First, let us compute the pairwise difference between the entire set of toxic embeddings and the entire set of non-toxic embeddings, $\mathbf{T} - \mathbf{N}$. This operation yields a set of difference vectors $\mathbf{d}$, with dimension $n \times$ *embedding-dim*. We can generate a set of coordinate vectors for toxicity by performing Principal Component Analysis (PCA) on the set of difference vectors $\mathbf{d}$, recalling that $\mathbf{d}$ is, in essence, how we define toxicity. The coordinate vectors are in fact the top-k eigenvectors that are found via principal component analysis.

Thus, principal component analysis is used to extract a set of eigenvectors V that constitute the toxic subspace. A sentence embedding $X \in \mathbf{T} \cup \mathbf{N}$ is projected onto the eigenvectors, giving a "toxic component." Subtracting the toxic component from the sentence embedding gives a residual vector that is the straightline distance from the toxic subspace to the original sentence.

$$X' = X - \langle X, V \rangle V \tag{3.1}$$

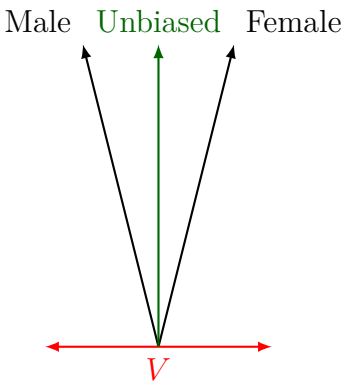

Figure 3.1: Illustration of subspace removal

By definition, this residual vector is orthogonal to the toxic subspace.

As show in Figure 3.1, we note the parallels between encoder detoxification and gender debias. We use the approach devised by Liang et al. (2020) to derive eigenvectors from the pairwise differences and to calculate the residual vectors.

Chapter 4

Experiments

4.1 Datasets

We train our approach on a combination of the following datasets: Jigsaw Toxicity Wulczyn, Thain, and Dixon 2017, Jigsaw Unintended Bias in Toxicity Borkan et al. 2019, OLID Zampieri et al. 2019, and Hate Speech and Offensive Language Davidson et al. 2017. As a preprocessing step, we split all examples in each dataset into individual sentences. We found that our dataset contained many examples that were multiple sentences in length, constituting a "long-text" classification problem. Since long-text classification is still very much an open research area, we sidestep this problem by analyzing sentence-level data. After splitting each example into sentences, we must ensure that each individual sentence matches the original label. For non-toxic example, we can assume this to be already true, but for toxic sentences, we cannot. To identify which sentences in toxic examples are toxic and which sentences are not, we use the toxicity classifier created by Dale et al. (2021) to filter out non-toxic sentences. Afterwards, we balance each dataset by undersampling before concatenating the results together to form our experimental dataset. From this experimental dataset we extract a test set and validation set each with a 50-50 distribution of 2000 toxic sentences plus 2000 non-toxic sentences, leaving a training set with 154604 sentences. We provide more details on each dataset further below.

4.1.1 Wikipedia Toxicity Subtypes

The Wikipedia Toxicity Subtypes (Jigsaw Toxicity) is a collection of human annotated comments from Wikipedia Talks archives provided by Wulczyn, Thain, and Dixon (2017). Each comment has a label 0 or 1, where 0 indicates a majority of annotators found the comment non-toxic, and 1 indicates a majority of annotators found the comment toxic. Balancing via undersampling yields a dataset with 30588 examples (prior to sentence-splitting).

4.1.2 Civil Comments

The Civil Comments dataset is a set of human annotated comments from the Civil Comments platform provided by Borkan et al. (2019). Each comment has a toxicity score on a scale from 0 to 1, representing the fraction of annotators that considered a comment toxic. We extract comments with a toxicity score greater than .75 and comments with a toxicity score exactly equal to 0, since scores in between may indicate disagreement among evaluators regarding toxicity. Balancing via undersampling yields a dataset with 74820 examples (prior to sentence-splitting).

4.1.3 OLID

The Offensive Language Identification Dataset (OLID) is a dataset of 14200 human annotated tweets Zampieri et al. (2019). The tweets consist of 4400 toxic labels, and therefore balancing via undersampling yields a dataset with 8800 examples (prior to sentence-splitting).

4.1.4 Hate Speech and Offensive Language

The Hate Speech and Offensive Language dataset is a dataset of 25000 human annotated tweets Davidson et al. (2017). Balancing via undersampling yields a dataset with examples (prior to sentence-splitting).

4.2 Toxic Subspace

We perform two experiments to determine how well out choice of coordinate system can explain toxicity. Recall that our coordinate system is the set of top-k eigenvectors returned by the principal component analysis. The first test is thus a verification of the principal component analysis itself. The second test examines how well the coordinate system can separate non-toxic and toxic training examples.

Because our method of calculating the toxic subspace is ultimately dependent on the choice of encoder, we test our approach on several different encoders. We choose as encoders BERT (Devlin et al. 2018), RoBERTa (Liu et al. 2019), and as a proof of concept, the encoder from the T5 sequence-to-sequence model (Raffel et al. 2019). For each encoder, we compare results from using the out-of-the-box pre-trained version against a more task specific version fine-tuned on the toxicity classification task (using the dataset described in §4.1). Intuitively, using the pre-trained encoder provides an embedding that is representative of the entire English language, while the fine-tuned encoder provides an embedding that highlights the toxic features.

4.2.1 PCA Scree Plots

An indication of the success of principal component analysis is a convergence of the eigenvalues as we examine less and less "principal" principal components. This phenomenon is often referred to as an "elbow curve" when the eigenvalues are plotted in order of magnitude in a scree plot. We provide the scree plots for each encoder in Figure 4.1, with the plots for both fine-tuned and out-of-the-box versions in the same figure. Note that, in order to plot both sets of eigenvalues in the same figure, we report the explained variance ratio, where the eigenvalue for a principal component is divided by the sum of all eigenvalues.

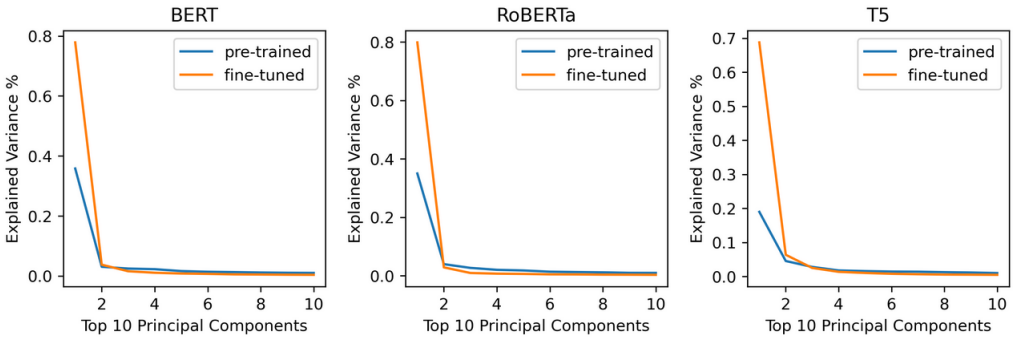

Figure 4.1: Scree plots for BERT, RoBERTa, and T5.

The scree plots in Figure 4.1. indicate that the PCA is able to find *some* subspace. We show in the next section that this subspace is indeed the toxic subspace. Additionally, we find that the rate of convergence for fine-tuned models is much faster than for the pre-trained models. Intuitively, this suggests that the fine-tuned encoder emphasizes toxic features, such that when the difference between toxic and non-toxic embeddings are calculated, there is much less noise.

4.2.2 Coordinate System

We have shown that PCA can find *some* subspace based on our data. Here, we show that this subspace is indeed the toxicity subspace. Let V be the top-2 eigenvectors produced by PCA (dimension = 2 $\times$ *embedding-dim*). We project S, a set of 2000 toxic and 2000 non-toxic examples (dimension = 4000 $\times$ *embedding-dim*) onto the 2 eigenvectors. The resulting operation SV^T yields a 4000 $\times$ 2 dimensional matrix, or in other words, a pair of coordinates for each sentence corresponding to the top 2 principal components. We use these coordinates to plot each sentence in Figure 4.2.

The results from Figure 4.2 illustrates that the coordinate system we have chosen can separate toxic and non-toxic examples for all encoders explored. Even in the case of the out-of-the-box encoders, which admittedly require more than two principal components to explain toxicity, there still appears to be some separation between the classes. By separating the toxic and non-toxic texts in the latent space, the coordinate system we use can define the toxic subspace.

4.3 Toxicity Removal

We test our proposed method of detoxification using the toxic subspace we have previously obtained. Our goal is to make all toxic embeddings non-toxic. If a toxicity classifier initially classifies a sentence as toxic, once we apply our detoxification step, the same classifier should now classify the sentence as non-toxic. In other words, one expected outcome of our approach is to induce low classification recall. At the same time, our detoxification step should not cause the classifier to alter its predictions about non-toxic sentences.

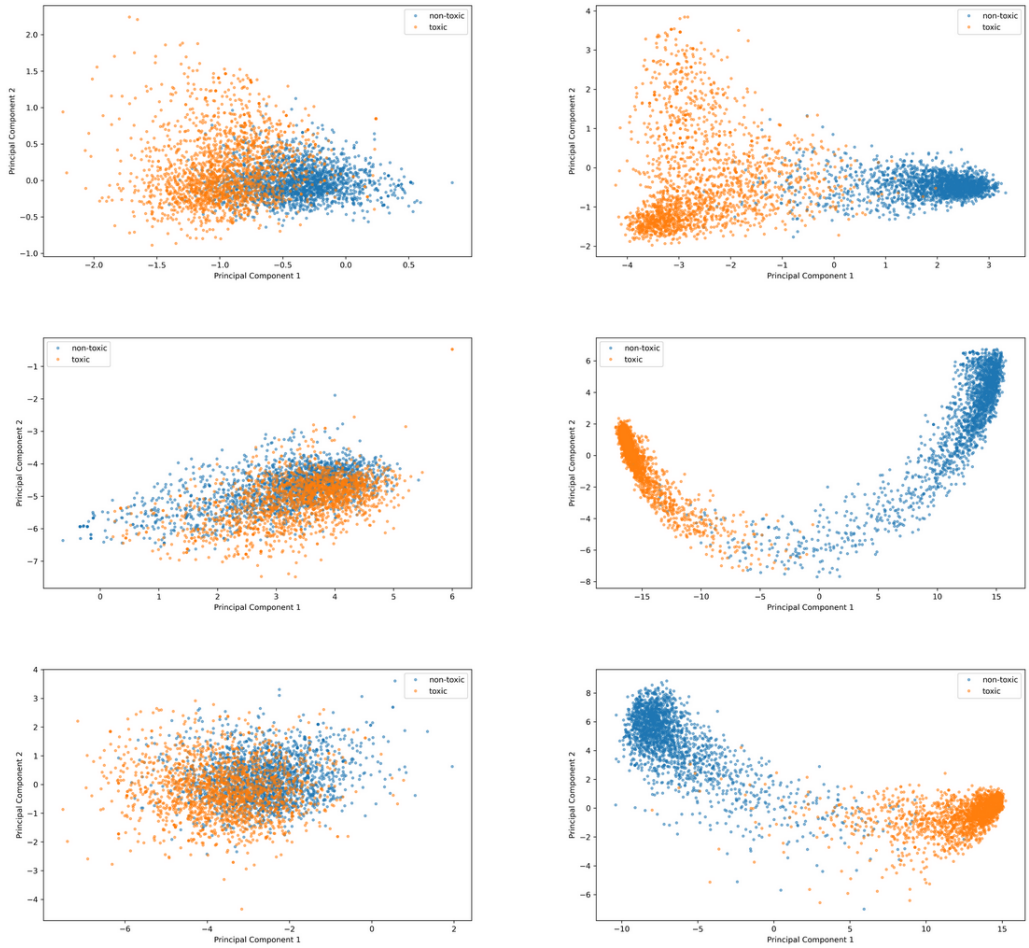

Figure 4.2: Toxic (orange) and non-toxic (blue) examples for T5 (top), RoBERTa (middle), and BERT (bottom) plotted along top 2 principal components.

4.3.1 Evaluation Models

We train a toxicity classifier specifically for the purposes of testing our detoxification approach. The toxicity classifier consists of an encoder with a classification head. The classification head takes as input the pooled embeddings from the encoder's last hidden layer output. We add a "backdoor" in between the encoder and classification head, where we can manipulate the encoder outputs per our detoxification method before passing the modified embeddings to the classifier head (Figure 4.3). We apply

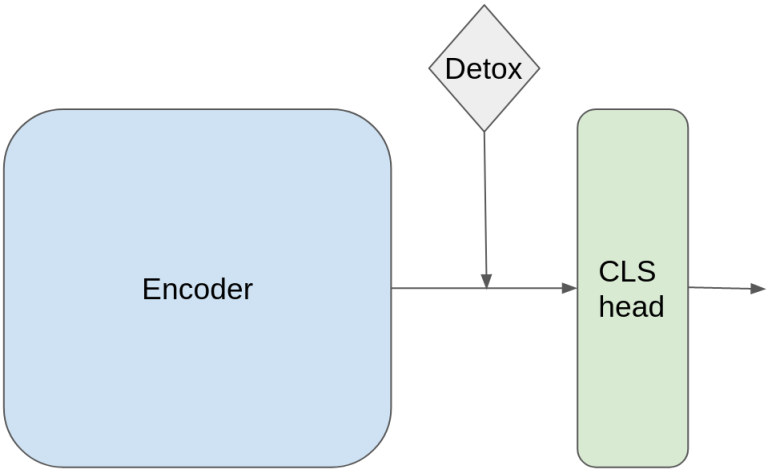

Figure 4.3: Evaluation model schematic diagram.

this framework on fine-tuned and out-of-the-box BERT, RoBERTa, and T5 models, yielding a total of six models.

4.3.2 Metrics

We define k to be the top-k principal components selected for the subspace in the detoxification step. To determine the extent to which our approach removes toxicity, we calculate 1) the *classification accuracy* (ACC), 2) the decrease in *cosine-similarity* between toxic embeddings and their corresponding detoxified embeddings (T-SIM), and 3) the number of toxic sentences predicted by the model ($\propto$ *recall*) before and after the detoxification step. To explore the extent to which our approach modifies the original embedding, we measure the decrease in *cosine-similarity* between non-toxic embeddings and their corresponding detoxified embeddings (N-SIM).

4.3.3 Results

Model	k	ACC	REC	T-SIM	N-SIM
BERT	3	0.6383	0.2850	$0.9062_{0.05}$	$0.9467_{0.03}$
RoBERTa	3	0.5000	0.0000	$0.8257_{0.02}$	$0.8325_{0.02}$
T5	15	0.5225	0.0460	$0.827_{0.05}$	$0.894_{0.04}$
BERT ft	2	0.5000	0.0000	$0.6356_{0.13}$	$0.8251_{0.08}$
RoBERTa ft	2	0.5000	0.0000	$0.7067_{0.06}$	$0.7704_{0.09}$
T5 ft	5	0.6698	0.3570	$0.7553_{0.10}$	$0.8123_{0.08}$

Table 4.1: Detoxification results.

	ACC	REC
BERT	0.8870	0.8860
RoBERTa	0.8608	0.9365
T5	0.9043	0.9315
BERT ft	0.9718	0.9865
RoBERTa ft	0.9723	0.9995
T5 ft	0.9665	0.9835

Table 4.2: Test set performance without detoxification.

We report the performance of each classifier on our balanced test set before and after subspace removal in Table 4.1. The results are reported in two groups, one for out-of-the-box models and another for fine-tuned models (ft). In the latter group, for the sake of consistency in the detoxification step we ensure that each subspace explains 80% of the model variance, roughly the amount of variance explained by two principal components. In the former group, because the eigenvalues of the subspace used in the detoxification step converge at a much slower rate, we ensure that each subspace explains at least 40% of the model variance. In the cosine-similarity columns, we report the averaged cosine similarity value along with the standard deviation. For ACC and recall, we present the results for the original encoder in Table 4.2 and the detoxified encoder in Table 4.1.

Across the different encoder models, our subspace removal algorithm is able to decrease the recall, accuracy, and before-and-after toxic embedding cosine similarity, while managing to preserve more of the non-toxic embedding. These results suggest that our approach is able to remove toxicity from encoders.

Chapter 5

Conclusions

In this paper we create a method to identify an underlying toxic subspace within pre-trained language model embeddings. Additionally, we propose a synthetic parallel corpus creation method to dynamically mask toxic lexical markers. Through our experiments, we provide evidence illustrating the validity of our subspace as a subspace for toxicity. Furthermore, we show that the subspace we find drastically decreases the amount of toxic predictions. This evidence strongly indicates that we have created a method to find and remove the toxic component from pre-trained encoder output.

5.1 Future Works

We consider how our toxic component removal method might be applied in a text generation setting. Suppose we have a sequence-to-sequence text generation model such as T5, with an encoder that feeds its final hidden layer output to a decoder. Crucially, this input to the decoder is *not* pooled, or in other words, the decoder receives token level input. Our largest obstacle rests in translating our detoxified pooled sentence-level embeddings back to individual token-level embeddings. However, because we use mean-pooling as our pooling strategy, we can use algebraic tricks to ”un-pool” our detoxified sentence level embeddings.

$$\text{Let } X \in \mathbf{T} \cup \mathbf{N}, V \text{ be the toxic subspace}$$

$$X' = X - XV^TV$$

$$\frac{1}{n}\sum_{i=1}^{n} t_i' = \frac{1}{n}\sum_{i=1}^{n} t_i - \frac{1}{n}\sum_{i=1}^{n} t_i V^T V$$

$$t_i' = t_i - t_i V^T V$$

In short, we can apply the same detoxification step on the token level using the same subspace as before.

5.2 Ethical Considerations

While language models grow increasingly powerful, they remain capable of generating offensive or extremist texts (Wallace et al. 2019; Gehman et al. 2020). Thus, they can pose a threat in the hands of bad actors, limiting their real world use (McGuffie and Newhouse 2020). Therefore, to mitigate the perpetuation of extremism, it is necessary to develop ways to prevent language models from learning toxic language.

Additionally, we are well aware that our dependence on lexical markers as an indicator of toxicity may lead to problematic model behavior. To address this concern, we identify lexical markers using the debiased toxicity classifier produced by Dixon et al. (2018) in the hopes that such a model will make unbiased decisions when identifying lexical markers.

Bibliography

Bolukbasi, Tolga et al. (2016). "Man is to Computer Programmer as Woman is to Homemaker? Debiasing Word Embeddings". In: *CoRR* abs/1607.06520. arXiv: 1607.06520. URL: http://arxiv.org/abs/1607.06520.

Davidson, Thomas et al. (2017). "Automated Hate Speech Detection and the Problem of Offensive Language". In: *CoRR* abs/1703.04009. arXiv: 1703.04009. URL: http://arxiv.org/abs/1703.04009.

Wulczyn, Ellery, Nithum Thain, and Lucas Dixon (2017). "Ex Machina: Personal Attacks Seen at Scale". In: *Proceedings of the 26th International Conference on World Wide Web*. WWW '17. Perth, Australia: International World Wide Web Conferences Steering Committee, pp. 1391–1399. ISBN: 9781450349130. DOI: 10.1145/3038912.3052591. URL: https://doi.org/10.1145/3038912.3052591.

Devlin, Jacob et al. (2018). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *CoRR* abs/1810.04805. arXiv: 1810.04805. URL: http://arxiv.org/abs/1810.04805.

Dixon, Lucas et al. (2018). "Measuring and Mitigating Unintended Bias in Text Classification". In.

Borkan, Daniel et al. (2019). "Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification". In: *CoRR* abs/1903.04561. arXiv: 1903.04561. URL: http://arxiv.org/abs/1903.04561.

Liu, Yinhan et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach". In: *CoRR* abs/1907.11692. arXiv: 1907.11692. URL: http://arxiv.org/abs/1907.11692.

Manzini, Thomas et al. (2019). "Black is to Criminal as Caucasian is to Police: Detecting and Removing Multiclass Bias in Word Embeddings". In: *CoRR* abs/1904.04047. arXiv: 1904.04047. URL: http://arxiv.org/abs/1904.04047.

Raffel, Colin et al. (2019). "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *CoRR* abs/1910.10683. arXiv: 1910.10683. URL: http://arxiv.org/abs/1910.10683.

Wallace, Eric et al. (2019). "Universal Adversarial Triggers for NLP". In: *CoRR* abs/1908.07125. arXiv: 1908.07125. URL: http://arxiv.org/abs/1908.07125.

Zampieri, Marcos et al. (2019). "Predicting the Type and Target of Offensive Posts in Social Media". In: *Proceedings of NAACL*.

Gehman, Samuel et al. (2020). "RealToxicityPrompts: Evaluating Neural Toxic Degeneration in Language Models". In: *CoRR* abs/2009.11462. arXiv: 2009.11462. URL: https://arxiv.org/abs/2009.11462.

Liang, Paul Pu et al. (July 2020). "Towards Debiasing Sentence Representations". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 5502–5515. DOI: 10.18653/v1/2020.acl-main.488. URL: https://aclanthology.org/2020.acl-main.488.

McGuffie, Kris and Alex Newhouse (2020). "The Radicalization Risks of GPT-3 and Advanced Neural Language Models". In: *CoRR* abs/2009.06807. arXiv: 2009.06807. URL: https://arxiv.org/abs/2009.06807.

Ravfogel, Shauli et al. (July 2020). "Null It Out: Guarding Protected Attributes by Iterative Nullspace Projection". In: *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, pp. 7237–7256. DOI: 10.18653/v1/2020.acl-main.647. URL: https://aclanthology.org/2020.acl-main.647.

Roller, Stephen et al. (2020). "Recipes for building an open-domain chatbot". In: *CoRR* abs/2004.13637. arXiv: 2004.13637. URL: https://arxiv.org/abs/2004.13637.

Song, Kaitao et al. (2020). "MPNet: Masked and Permuted Pre-training for Language Understanding". In: *CoRR* abs/2004.09297. arXiv: 2004.09297. URL: https://arxiv.org/abs/2004.09297.

Wang, Wenhui et al. (2020). "MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers". In: *CoRR* abs/2002.10957. arXiv: 2002.10957. URL: https://arxiv.org/abs/2002.10957.

Dale, David et al. (Nov. 2021). "Text Detoxification using Large Pre-trained Neural Models". In: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. Online and Punta Cana, Dominican Republic: Association for Computational Linguistics, pp. 7979–7996. DOI: 10.18653/v1/2021.emnlp-main.629. URL: https://aclanthology.org/2021.emnlp-main.629.

Laugier, Leo et al. (2021). "Civil Rephrases Of Toxic Texts With Self-Supervised Transformers". In: *CoRR* abs/2102.05456. arXiv: 2102.05456. URL: https://arxiv.org/abs/2102.05456.

Zhou, Xuhui et al. (2021). "Challenges in Automated Debiasing for Toxic Language Detection". In: *CoRR* abs/2102.00086. arXiv: 2102.00086. URL: https://arxiv.org/abs/2102.00086.

Appendices

Appendix A

Implementation Details

PCA was calculated using the sklearn library. Models were written with PyTorch. For the tokenizer, we used a sequence length of 128. When creating the parallel corpus, we dropped sentences greater than 64 tokens in length and sentences with more than a quarter of tokens masked due to hardware constraints. For each toxicity classifier, we used a learning rate of .001 and stochastic gradient descent with 5 epochs. We used an additional 80-20 split on the toxicity dataset while training the toxicity classifiers. Model results are reported after one run with random seed set to 42. Our experiments were computed using Google Colab's GPU backend (16 GB GPU, 25 GB RAM, 147 GB Disk).