Detoxification of pre-trained Language Models through Subspace Transformation

Mohit Sudhakar

Department of Computer Science

University of Virginia

Charlottesville, VA 22904

ms5sw@virginia.edu

Submitted to the Department of Computer Science on April 28 2021, in partial fulfillment of the requirements for the degree of

Master of Science in Computer Science

at the

UNIVERSITY OF VIRGINIA

May 2021

Abstract

Large pre-trained language models have achieved significant performance on a multitude of language understanding and inference tasks. However, because of training on large-scale unfiltered texts on the internet, they often encode some unexpected information, such as toxic or abusive language, which makes the real-world usage of these language models limited. Our work investigates the existence of a low-dimensional bias subspace in the latent space of pre-trained language models. Empirically, we show that there exists a common bias subspace across four popular models, BERT, GPT-2, RoBERTa, and XLNet, and conduct layer-wise subspace analyses of these models. We provide a method to construct a bias subspace by manipulating principal components with linear transformations. Our debiasing method does not require fine-tuning and can be used on existing pre-trained language models at inference. We also provide a methodology for constructing parallelized datasets using word-level bias scores. This method can be used for constructing parallel datasets for any kind of bias. We show that when the bias subspace is removed, the toxic sentence classification accuracy drops, and this result is consistent across pre-trained models like BERT and other variants.

Contents

List of Figures

List of Tables

List of Algorithms

1 Introduction

1.1 What is the problem studied and why it is essential?

The field of natural language processing has evolved at an astonishing pace in the past few years. The transformer (Vaswani et al. (2017)) has transformed NLP and improved performance on many tasks. Bigger models with multiple transformer blocks have resulted in state-of-the-art performance. However, neural language models are often considered and used as black boxes in various tasks. In order to understand latent representations learned by models, make them interpretable and enable their usage in conservative areas like finance, we need methods that explore how a model works and try to find and fix so-called bugs in the model. Pre-training and fine-tuning methods help us obtain a generalized representation of the data. The resulting high dimensional latent space can then be fine-tuned for a specific task with more data. Although pre-training is a critical function and encodes many desirable properties of language in the model, it requires a large amount of data. As such, most internet text data used in pre-training end up containing biases. Biases related to gender, race, religion, and content are essential to explore since they affect the output generated by the language model.

We study the problem of content bias, specifically toxicity, which includes abusive language and harmful content. We focus on this type of bias, as in our opinion, this bias seems to affect real-world adoption most of all. Gehman et al. (2020) have also pointed out that neural language models can generate toxicity without any prompt or training. As a specific example, consider a writing assistant application that suggests ideas for starting a new paragraph based on previous paragraphs. People would not want to use a writing assistant application that may produce harmful content, as it would be untrustworthy to produce quality output. Modern large pre-trained language models have many potential applications that would not be useful if they were to produce such biased content.

1.2 The current progress in this direction and what is missing?

The majority of current research related to understanding and controlling biases in language models fall into three categories, i.e., controllable text generation using bias control objectives, adversarial generation using adversarial examples, and de-biasing model embeddings using subspace manipulation techniques. These methods are briefly described in Section 2. Such methods require a lot of training data and fine-tuning with hyperparameter tuning to get right, which is not practical for researchers with low resources. Fine-tuning a large pre-trained language model requires a non-trivial amount of computational resources. Additionally, these methods also do not address the issue in the pre-trained model itself. However, there is little recent work on understanding the internal workings and issues of pre-trained models.

1.3 The Proposed Study

To get a deeper understanding of the various pre-trained models, we try to construct a subspace for toxicity and explore it at a layer-wise (local) level and a global level, where we consider the model without fine-tuning. In this work, the first step is to understand if there exists a subspace that encodes toxicity in large pre-trained language models. We confirm this hypothesis by creating a toxicity classifier that gives good accuracy. Next, we remove the principal components of this subspace using linear transformations with the high dimensional latent space and build classifiers on the resulting de-biased model to show that the resulting classifier has a significant drop in accuracy. We create a parallel dataset for toxicity consisting of toxic and non-toxic sentences and demonstrate it on the RealToxicityPrompts (Gehman et al. (2020)) dataset.

At the local level, we conduct the above experiment for BERT in a layer-wise manner. We observe that the trained classifier gives 85% accuracy for the biased model and 66% for the layer-wise de-biased variant of the BERT model. We also conduct analyses of the bias subspace created by 4 prominent models, BERT (Devlin et al. (2018)), GPT2 (Radford et al. (2019)), RoBERTa (Liu et al. (2019)) and XLNet (Yang et al. (2019)). Our methodology can be used for any pre-trained language

model. Lastly, we point out certain caveats of the local approach and address them in the global variant.

For the global variant, we create a bias subspace using incremental PCA and use this to de-bias the output of pre-trained language models. This experiment is run by removing varying sizes of the subspace to understand the approximate size of the subspace to remove to get a de-biased output. We observe that the model gets worse in predicting toxicity as more dimensions of the bias subspace are removed. Experiments are conducted with different transformer-based architectures using 4 prominent language models; BERT (Devlin et al. (2018)), RoBERTa (Liu et al. (2019)), ALBERT (Lan et al. (2019)) and DistilBERT (Sanh et al. (2019)) and show that our observations hold across all models.

1.4 Summary of Contributions

- We provide a method to construct a bias subspace by manipulating principal components with linear transformations.
- We analyze the subspace generated by various large pre-trained language models.
- We provide a methodology to create parallel datasets from existing classification datasets.
- We build 20 classifiers to show accuracy drops in the de-biased models and conduct analyses for de-biased models for multiple subspace sizes. Our model de-biasing method can be used in an online setting for inference.

2 Related Work

Current methods for controlling biases in deep neural network models include work on controllable text generation, adversarial generation, and subspace exploration methods.

1. Previous work on biases using Controllable text generation

Sheng et al. (2020) presents an approach to control societal biases using 'regard' towards a demographic as a metric by using bias control objective functions. They use custom loss functions using linear combinations of attributes to guide the model into producing desirable outputs. Dathathri et al. (2019) propose Plug and Play Language models, a framework to control text generation in a specific direction, using gradients from discriminative attribute models.

2. Previous work on toxic output using Adversarial generation

Producing and defending against adversarial examples is an important topic in current NLP research. Gradient-based word substitution attacks (Ebrahimi et al. (2017) and adversarial triggers Wallace et al. (2019)) have been shown to produce harmful text. Michel et al. (2019) use meaning-preserving perturbations to improve adversarial training and evaluation of sequence-to-sequence models.

3. Previous work on Subspace exploration

The existence of a subspace for gender bias in word embeddings was first shown by Bolukbasi et al. (2016). Reif et al. (2019) to measure and visualize the geometry of the latent space produced by BERT. A variety of biases have been explored, such as subjective bias by Pryzant et al. (2020). Bhardwaj et al. (2020) and Liang et al. (2020) where gender and other social biases in BERT are investigated and removed using PCA. However, they focus only on BERT and do not show results on other popular models. Our methodology for removing bias extends their approach of layer-wise bias removal and applies it to toxicity removal. We point out certain caveats with this method and propose a 'global' way of de-biasing, without fine-tuning of

any kind. We show that our method works in real-world scenarios and can be used for inference.

3 Methodology

In this section, we intuitively describe our methodology for exploring if a bias subspace exists and its removal from the latent space of the model using a linear transformation. We also present our process for parallel dataset creation.

3.1 Exploring Subspace

We hypothesize that there exists a low-dimensional subspace that encodes bias information in most pre-trained models. In order to explore and determine if a toxic subspace exists, we train a classifier to perform sentence classification for toxicity detection. The intuition is that a high accuracy by a toxicity classifier trained on a pre-trained language model would show that the model is learning to predict toxic sentences correctly, which signifies that there exists a part of the model that encodes information related to toxicity.

For training all classifiers, we use the RealToxicityPrompts dataset from AI2 Gehman et al. (2020). This dataset consists of sentences with a corresponding toxicity score. The dataset provides multiple scores for varying toxicity levels. We consider the 'severe_toxic' label as the toxicity score for the sentence, with a score of 0.5 or greater as toxic and remaining as non-toxic. We leave the exploration using other types of scores to future work.

We build a classifier on Google's BERT, and the performance is shown in Figure 1. The following hyperparameters are used for this classifier. The classifier is run using stochastic gradient descent for ten epochs, learning rate of 0.0001, a sequence length of 128, and batch size of 32. We get a high accuracy of 85%, which confirms our hypothesis that a subspace could exist. BERT parameters are not frozen and allowed to change, which causes the model to learn good predictions.

Next, we describe our method for finding the basis vectors for the subspace by using PCA to extract principal component directions of the low-dimensional subspace.

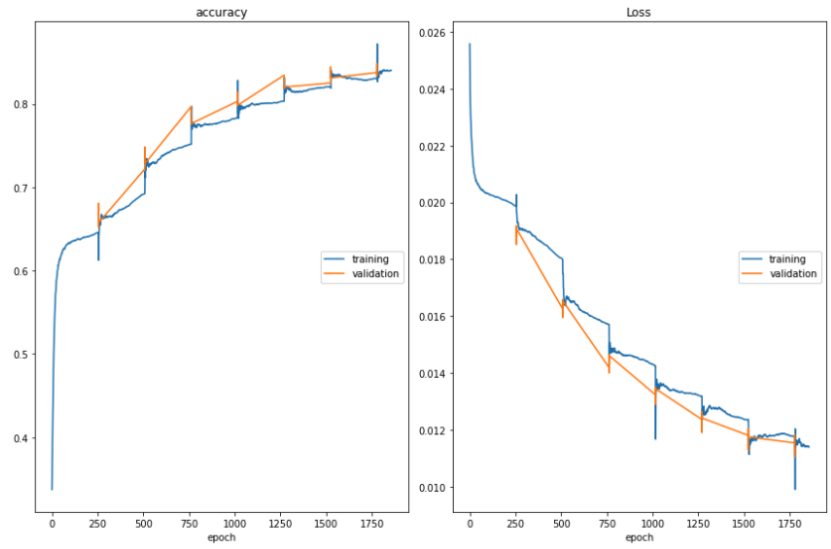

Fig. 1: Normal (biased) BERT - Accuracy 85%

3.2 Finding Subspace using PCA

[1] **Intuition**: Figure 2 (left) provides an intuitive understanding of the method for finding a direction for toxicity. We consider two vectors; the first is a toxic word or sentence vector E_t, which contains toxicity and other components encompassing other language features. The second is its corresponding non-toxic vector E_{nt}, which would presumably not contain any toxicity information. Then, their difference vector $D = E_t$ - E_{nt} gives us the direction of toxicity for that word or sentence. The actual direction vector will be high-dimensional, but for visualization, Figure 2 (right) shows the two-dimensional scatter plot of the direction vector and PCA being performed on it. We see that the direction of toxicity is given by the principal component with the highest explained variance.

With this intuition in mind, we consider a toxic sentence S_t, its corresponding non-toxic sentence S_{nt}, and our model M. The non-toxic sentence should be such that all the words except the toxic word should be the same. We assume that most of the information about toxicity is encoded in one or a few words. For example, if we have

[1] This section may contain harmful or toxic language. Reader caution advised.

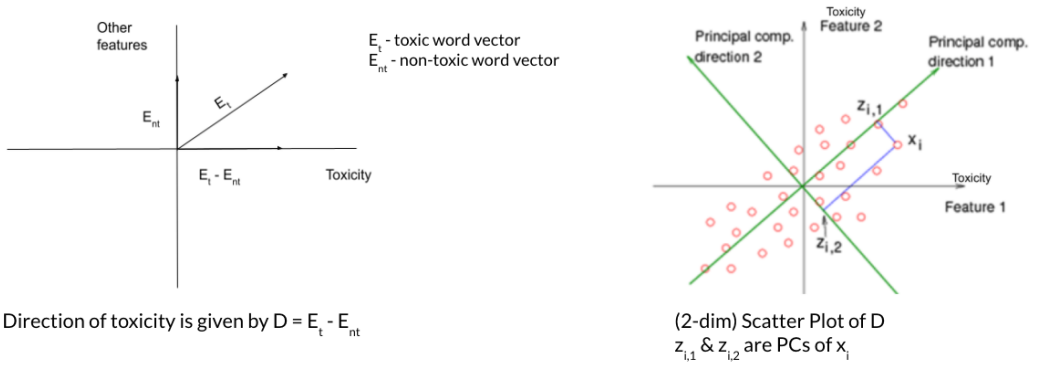

Fig. 2: Intuition of Toxicity Direction and Principal Components

a toxic sentence like, "That is f***ed up.", then the corresponding non-toxic version could be "That is <PAD> up.". We replace the toxic word with the corresponding PAD token of the model. The reason for using the PAD token is two-fold. First, in toxicity, unlike gender bias, it is challenging to find the appropriate non-toxic word for any given toxic word, and the PAD token is used as it does not encode any unwanted information. Second, the PAD token comes in handy while performing attention on the input sentence. Generally, we input an attention mask to pre-trained language models to signify that padded sentences should attend only to the words and not the padding tokens. We attend to the padding token in our experiment by using the same attention mask as the toxic sentence. Note that only the toxic word changes and remaining words are the same.

Figure 3 describes our methodology for obtaining the principal component directions representing the basis vectors of the low dimensional bias subspace. First, we tokenize sentences S_t and S_{nt} to get T_t and T_{nt}. Then pass the tokenized inputs to M to get corresponding toxic and non-toxic encoded outputs E_t and E_{nt}. To find the direction of toxicity in the latent space that the encoded output exists in, we take the difference of E_t and E_{nt}. This method allows us to nullify all the other words and only concentrate on the toxic words. The resulting difference vector should contain information about the principal direction of toxicity in the model (for that batch of sentences). Next, we perform PCA on the difference vector to get the principal component matrix. This matrix encodes the subspace that we will use to

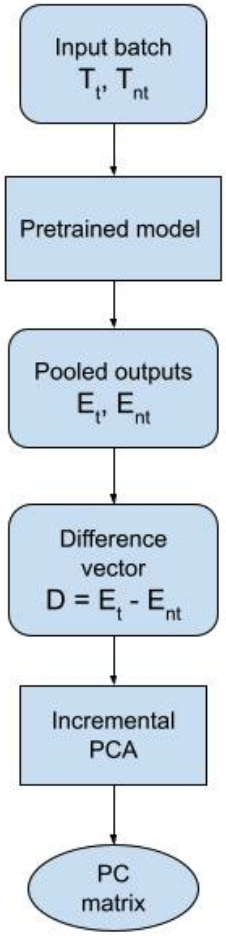

Fig. 3: Principal Component Matrix Creation

de-bias our model output. PCA is used because our task is to find a low-dimensional subspace that encodes toxicity information. Using PCA, we can manipulate and experiment with the size of the subspace that we want to remove. Also, PCA is a linear dimensionality reduction method and is faster and less computationally intensive than other non-linear methods. Although other methods could also be used, this could affect the inference time taken by the model.

3.3 Remove Subspace Through Linear Transformation

Now that we have the bias subspace represented by the principal component matrix, we perform a linear transformation to remove those components from the encoded output. Equation 1 describes the function RC that is used to remove the components.

We calculate the dot product of E_t and P to get the component in the bias direction. We multiply this by P to get the perpendicular projection of E_t to P and remove it from our original encoded output E_t.

$$RC(E_t, P) \leftarrow E_t - \langle E_t, P \rangle \cdot P \tag{1}$$

P [number of components x embedding dimension] is the principal component matrix. The number of components is a hyperparameter that determines the dimension of the subspace to be removed. The higher the number of components, the greater the 'size' of the subspace. The embedding dimension is the model's embedding dimension (generally 768 for models like BERT, GPT2, and variants). E_t [batch size x embedding dimension] is the pooled model output of the toxic sentence. The pre-trained model output of shape (batch size, sequence length, embedding dimension) consists of information for each word. In the bidirectional transformer (BERT-based) models, the pooled output is obtained by taking the encoding corresponding to the first token in the sentence and provided by the Huggingface transformers library. For generative (GPT2-based) models, the pooled output is obtained by taking the encoding corresponding to the last word in the sentence. This operation is done similarly to the Huggingface transformers library.

3.4 Dataset Creation

The above methodology requires a parallel corpus of toxic and corresponding non-toxic sentences. However, no such large-scale dataset is publicly available. We build upon the recently released RealToxicityPrompts dataset Gehman et al. (2020) to build a parallel corpus described as follows. The RealToxicityPrompts dataset consists of 100,000 sentences with their corresponding toxicity scores obtained using Google's Perspective API. The label distribution 60:40, which means there are 60,000 toxic and 40,000 non-toxic sentences. We used stratified sampling for equal distribution of labeled data. Note that this dataset contains multiple toxicity scores; we use the one labeled 'severe_toxic'. We consider sentences with a score >= 0.5 as toxic (as per their paper). All future classifiers are also trained using this

dataset. Please note that for all pre-trained models, we use the base versions in the HuggingFace transformers library [2], which consist of 12 layers of transformers.

The steps for building a parallel dataset is described as follows;

1. Split the original dataset by labels into toxic and non-toxic datasets based on toxicity score.
2. Tokenize both datasets. To maintain consistency of tokens, this is done by using tokenizer of the respective models.
3. For each word, count all occurrences in toxic and non-toxic datasets.
4. Get word-to-sentence maps for toxic and non-toxic datasets. For each word, we get all sentences that contain this word.
5. Calculate toxicity percent for each word.

$$\text{toxicity percent for a word} = \frac{\text{occurrences in toxic sentences}}{\text{total occurrences}} \tag{2}$$

6. Sort words in decreasing order of toxicity percent

We perform additional preprocessing steps like removing low-frequency words ($<=$ 3) and select top-k toxic words and their sentences as data for learning the subspace

[2] https://huggingface.co/transformers/

4 De-biasing Pre-trained Models in a Layer-wise manner

4.1 Building a Layer-wise De-biased Model

We describe the model update methodology for obtaining a bias subspace in this section. By breaking down a large pre-trained model in a layer-wise fashion, we hope to understand the behavior of the bias subspace.

Algorithm 1: Layer-wise de-biasing algorithm

Input : Sentences, toxic S_t and non-toxic S_{nt}
Output : Final layer output u_k and principal components P_k
T_t = Tokenize(S_t)
T_{nt} = Tokenize(S_{nt})
u_0 = Layer$_0$(T_t)
v_0 = Layer$_0$(T_{nt})
D_0 = u_0 - v_0
P_0 = PCA(D_0)
for *Layer k := 1 to 12* **do**
  u^*_{k-1} = RC(u_{k-1}, P_{k-1})
  v^*_{k-1} = RC(v_{k-1}, P_{k-1})
  u_k = Layer$_k$ (u^*_{k-1})
  v_k = Layer$_k$ (v^*_{k-1})
  D_k = u_k — v_k
  P_k = PCA(D_k)
end

Algorithm 1 describes the process we use to obtain a layer-wise subspace and use Equation 1 to remove principal component directions. This algorithm is adapted from Bhardwaj et al. (2020) This algorithm takes as input a pair of toxic and its corresponding non-toxic sentence. First, we pass the tokenized inputs to the embedding layer Layer$_0$ of the pre-trained model. Then, we calculate the difference vector D_0 and use PCA to get the principal component subspace for the embedding layer. Similarly, this process is repeated for each subsequent layer of the model. The outputs obtained are the final layer de-biased output and the principal components calculated at each layer. We analyze the principal components obtained for four different pre-trained models in Section 4.

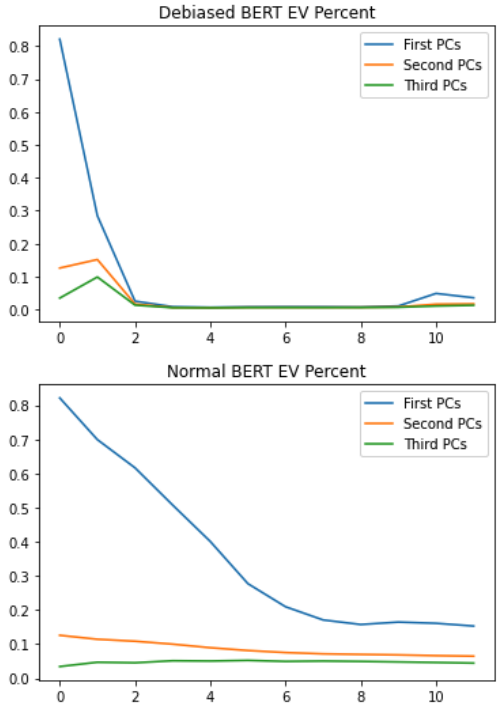

Fig. 4: BERT Explained Variance%

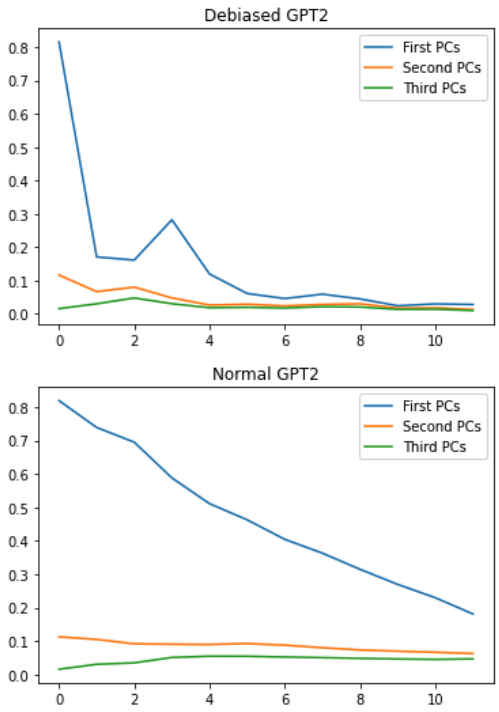

Fig. 5: GPT2 Explained Variance%

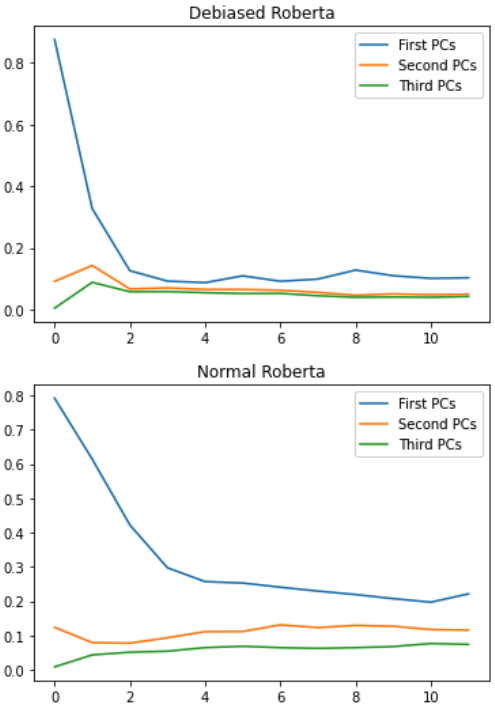

Fig. 6: RoBERTa Explained Variance%

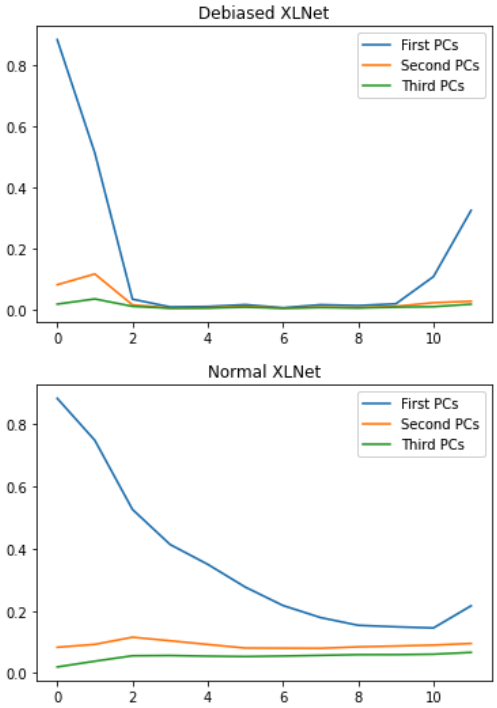

Fig. 7: XLNet Explained Variance%

4.2 Subspace Analysis

This section analyzes the subspace formed by four prominent pre-trained models - BERT, GPT2, RoBERTa, and XLNet, by plotting the principal components by decreasing order of explained variance. Explained variance is a measure of how much variance a particular dimension or feature has, which corresponds to how important the feature is. We show plots of top-3 principal components by explained variance for each layer. The X-axis represents Layers of the model; Y-axis represents % contribution of explained variance of principal component for that layer. Both biased and de-biased versions are shown.

We make four major observations from the plots shown in Figures 4 through 7. These plots show principal components obtained from the normal (biased) and de-biased versions of BERT, GPT2, RoBERTa, and XLNet models.

1. We observe that the first two principal components show significant values in the first layer and as we go deeper into the model, the explained variance decreases. This means that the subspace becomes more dispersed as it captures high-dimensional features with more layers, and this observation holds across all four models with varying degrees.

2. The de-biased model contributions are lower than their biased model counterparts, which is the expected result of the de-biasing methodology.

3. Changes in the observations of the biased models may correspond to the amount of data and training procedure used while pre-training. For example, for RoBERTa and XLNet, the contributions decrease quickly in early layers, which means that they learn high dimensional features earlier (w.r.t layer) than BERT or GPT2. This is in keeping with results that show RoBERTa and XLNet to perform better than vanilla BERT on GLUE tasks.

4. The last few layers may contain more toxicity information, as we see minor spikes in BERT, RoBERTa, and XLNet. This is despite our layer-wise de-biasing method being applied. We think this is because the final few

layers learn high dimensional features, including toxicity information, and reintroduce toxicity into the model to a small extent.

4.3 Training Local De-biased Classifier using BERT

We call our de-biased BERT model LocalDeBERT and train a toxicity classifier using LocalDeBERT to demonstrate the working of the layer-wise de-biasing algorithm. This version of BERT consists of the same parameters (12 layers) as the pre-trained BERT model. The output at each layer is de-biased using Equation 1 and sent to the next layer. The hyperparameters are similar to those mentioned in Section 3.1. The classifier is run using stochastic gradient descent for ten epochs, learning rate of 0.0001 and batch size of 32. The parameters of pre-trained BERT are not frozen and allowed to change, which allows the model to learn to predict toxic sentences. The training performance plot is shown in Figure 8. We get a low accuracy of 66%, which means the layer-wise de-biasing method works, and our model cannot predict toxic sentences correctly. This means the part of the model encoding toxicity is removed and that the performance of 66% could be from the fully connected layer of the classifier.

4.4 Discussion and Caveats

The performance drop obtained by LocalDeBERT proves our hypothesis that a bias subspace exists and that our layer-wise de-biasing method works. We have focused on toxicity in this thesis; however, this methodology can be applied to any kind of bias, with or without a parallel dataset.

Even though this method can be used for de-biasing models, the disadvantage of this method is that it cannot be used for inference since it requires changes to the model's layers. Hence the principal component matrix that we learn using PCA is only for a specific batch, and we do not have an overall PC matrix that can be used for inference. Additionally, this model cannot work in an online environment, as it requires us to have all the inputs from the beginning. We solve the inference problem in the next section.

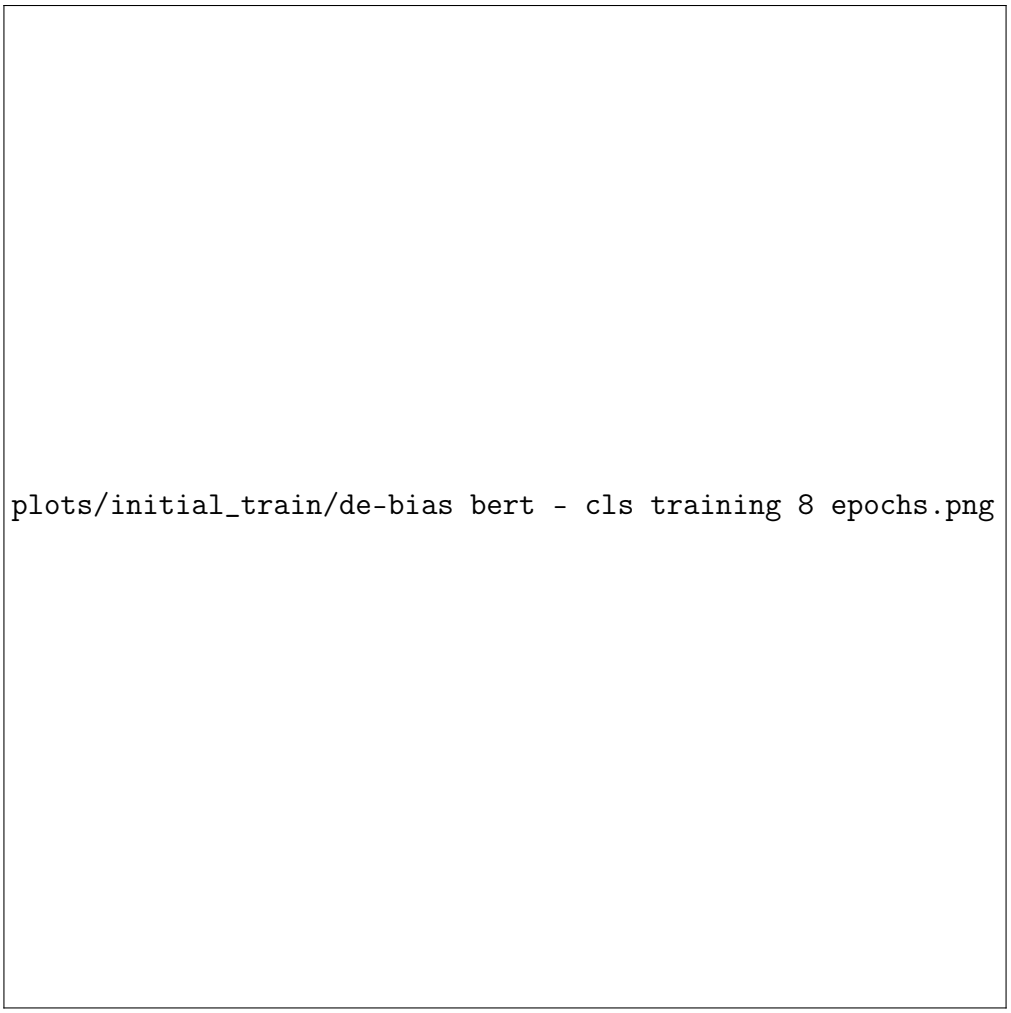

Fig. 8: LocalDeBERT: locally de-biased BERT - Accuracy 66%

5 Global De-biasing of Pre-trained Models

5.1 Global subspace using Incremental PCA

Models: We use BERT and 3 of its variants to test our methodology. They are described below. We leave the analysis of GPT2 and other pre-trained language models to future work.

- RoBERTa (Liu et al. (2019)) is a robustly trained version of BERT with more data and training time.
- ALBERT (Lan et al. (2019)) is a light variant of BERT with increased training speed and consuming less memory.

- DistilBERT (Sanh et al. (2019)) is a distilled variant of BERT with almost half the number of parameters. This allows inference to be much faster than BERT.

In this section, we create a subspace that encodes information from the entire dataset. Building such a subspace would allow us to de-bias models at runtime, i.e., during inference. We use the methodology described in Sections 3 and 4. Note that we used layer-wise de-biased versions of the pre-trained models in Section 4, however for global de-biasing, we do not make any changes to the pre-trained model. There are no fine-tuning or model changes to the underlying pre-trained model itself. The process is as follows: For a batch of sentence pairs S_t and S_{nt}, we get their tokenized inputs T_t and T_{nt} and pass them through the pre-trained model. Next, we take the pooled model outputs E_t and E_{nt} and calculate the difference vector D and run PCA on D. The Pooling process is described in Section 3.3. We use incremental PCA to approximate the subspace representing the entire bias subspace. The difference vectors D are passed through the same incremental PCA object for each batch of sentences. After running for the entire dataset, the output is a matrix of size (number of components, embedding dimension) and represents the basis vectors of the bias subspace. The final subspace matrices are stored in files and will be used during the inference step.

The dataset we use for this process is a subset of the dataset we described in Section 3.4. We use the sentences containing the top 100 toxic words according to frequency and toxicity score for all models. We noticed no significant changes in the explained variance of the principal components when more than the top 100 words were used.

This process is repeated for BERT, ALBERT, RoBERTa, and DistilBERT. For each model, a new subspace is constructed. In Figures 9 through 12, we see plots of principal components' explained variance for the first 10 PCs. The X-axis denotes the principal component dimension, and Y-axis denotes the contribution of the explained variance for that component. We observe that most of the variance is present in the first few principal components, which further confirms our initial hypothesis that a

low dimensional bias subspace exists. We do not show results for GPT2 since it is primarily a generative model and generally not used for classification. However, we plan to explore GPT2 and other variants as well and leave that to future work.

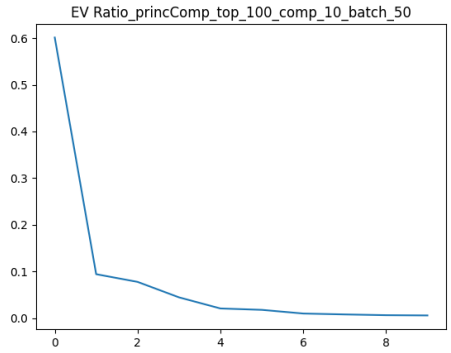

Fig. 9: BERT Explained Variance%

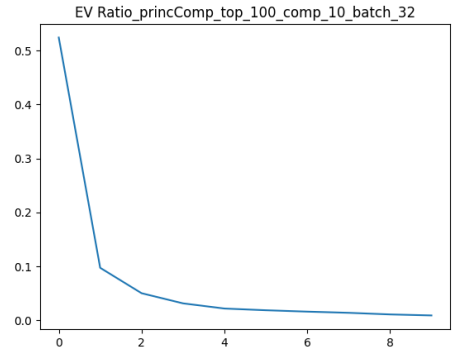

Fig. 10: ALBERT Explained Variance%

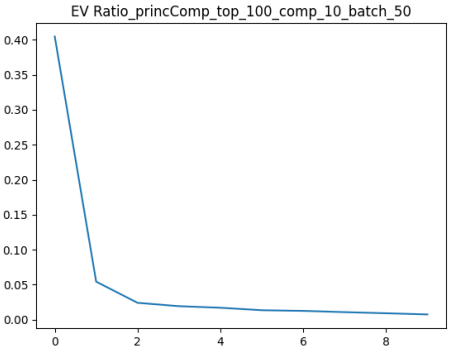

Fig. 11: RoBERTa Explained Variance%

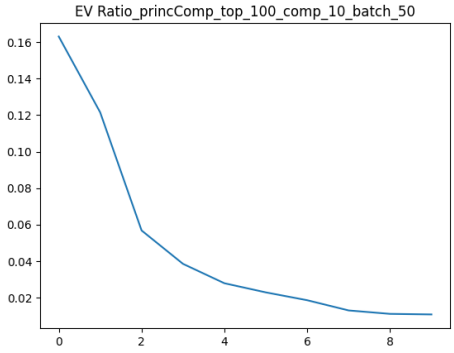

Fig. 12: DistilBERT Explained Variance%

5.2 Classifier and Inference

In this section[3], we describe our global model architecture, which uses the subspace we found in the previous section. Note that we do not make any changes to the underlying pre-trained model while performing inference. The outputs of the pre-trained model are de-biased by removing the projection as described in Section 3.3 and are then passed to a fully connected layer for classification. This is described in Figure 13. This method allows us to de-bias any model, provided the subspace creation is done as per the previous section.

[3]This section may contain harmful or toxic language. Reader caution advised.

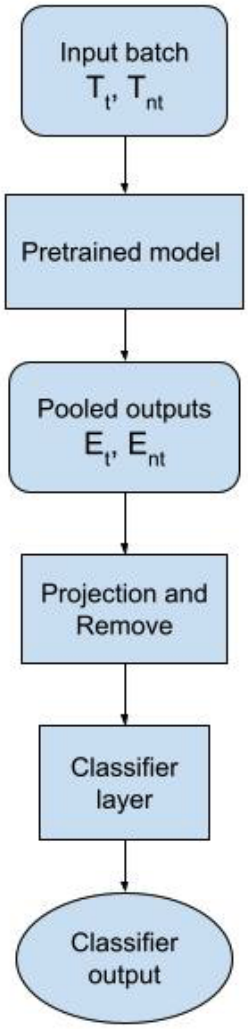

Fig. 13: Global Classifier

We show the de-biasing process for a sample toxic sentence in Table 1. As expected, the de-biased model loss is higher than the biased model loss, which implies that the de-biased model cannot be classified reasonably. However, the loss may vary across models; it is generally higher than the biased model's loss. Loss is compared instead of a metric like accuracy because the accuracy is not a true reflection of the success of the de-biasing process. Consider the softmax output of the classifier to be 0.7, which implies that the prediction for the input sentence should be 'toxic' since 0.7 is closer to 1. Now, after de-biasing, if the softmax output changes to 0.6, the sentence would still be classified as 'toxic', and the accuracy would remain unchanged. We

want to capture the uncertainty of the model in classifying a particular input, and hence the loss is chosen to be a metric for comparison.

Sentence: What the f*** is wrong with you? Is that even possible you idiot!		
Model	Biased Loss	de-biased Loss
ALBERT	0.262	**0.527**
BERT	0.537	**0.625**
RoBERTa	0.708	**0.817**
DistilBERT	0.684	**0.736**

Table 1: Inference with example sentence.

5.3 Training Classifiers

We train classifiers for the four models - BERT, ALBERT, DistilBERT, and RoBERTa, and describe the training process in this section. The hyperparameters used for all classifier models are the same as in Section 3. The classifier is run using stochastic gradient descent for ten epochs, learning rate of 0.0001, and sequence length of 128. We use a batch size of 32 for all models. The classifier layer is a linear layer of (embedding dimension x number of labels). The number of labels is 2; a sentence can be toxic or non-toxic. The parameters of all pre-trained models are frozen, and only the classification layer is allowed to change. This allows us to observe the model's behavior without changing the model parameters to improve accuracy (which would allow the model to learn a different subspace).

We train 20 classifiers in total, 5 for each pre-trained model. For a specific pre-trained model, we show the classifier's performance after removing four varying dimensions of the subspace and the original model itself. For each of BERT, RoBERTa, and DistilBERT, we remove the subspaces containing 5, 10, 25, and 50 dimensions. For ALBERT, we remove the subspaces containing 5, 10, 20, 32 dimensions. This change is due to GPU and hardware limits while performing computations.

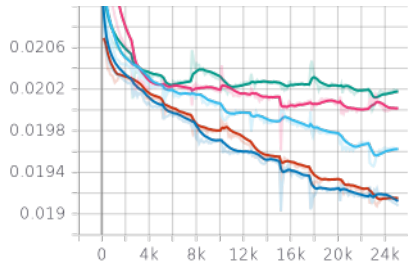

Fig. 14: *BERT Train Loss

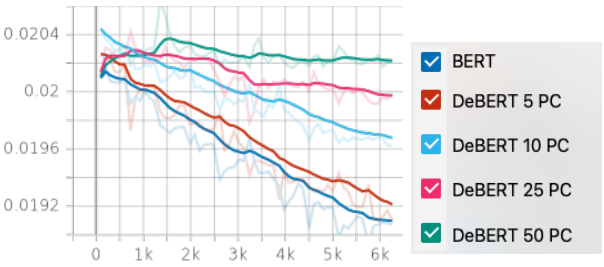

Fig. 15: *BERT Val Loss

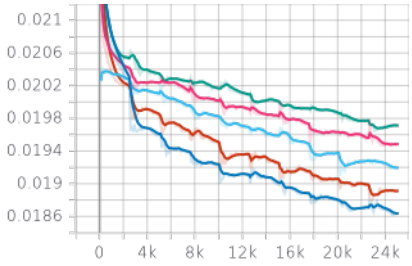

Fig. 16: *ALBERT Train Loss

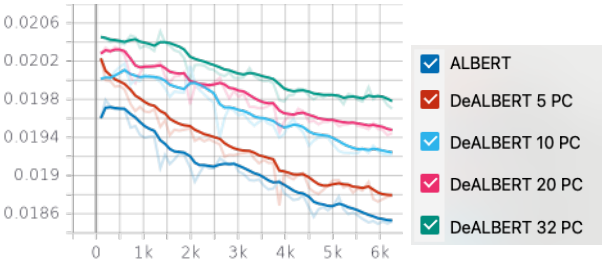

Fig. 17: *ALBERT Val Loss

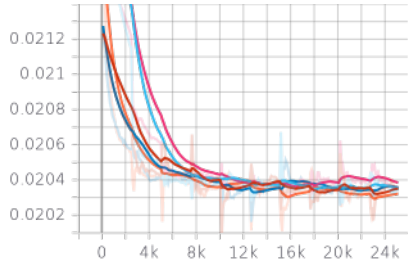

Fig. 18: *RoBERTa Train Loss

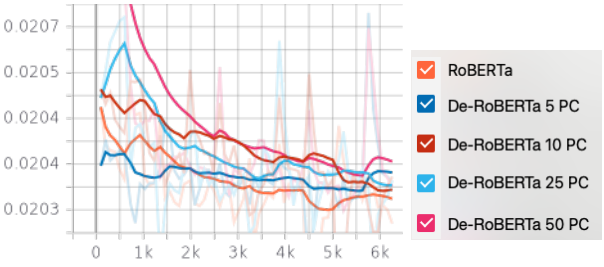

Fig. 19: *RoBERTa Val Loss

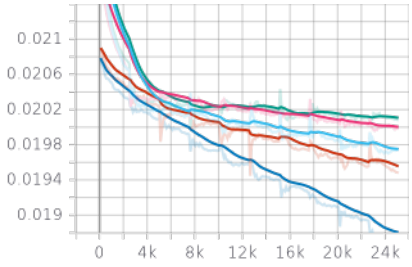

Fig. 20: *DistilBERT Train Loss

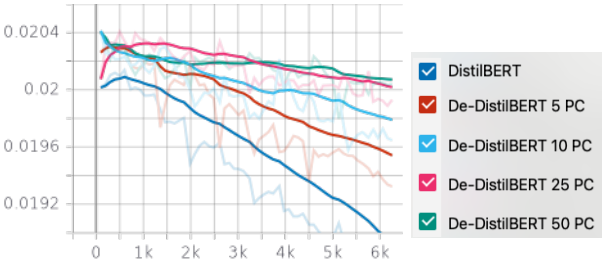

Fig. 21: *DistilBERT Val Loss

Figures 14 through 21 show plots for the training and validation loss for pre-trained models and their de-biased versions. For each model, the legend on the right of each plot denotes the number of principal components (PC) or the size of the subspace removed. We denote the de-biased versions of the model by adding 'De' to the model name.

6 Observations and Conclusions

6.1 Observations

We make two major observations from the plots shown:

- Both training and validation losses are lowest for the original models. This shows that the classifier trained on the original BERT performs the best at the toxicity classification task.

- The validation loss increases as the number of principal components removed are increased. This is especially true in BERT, ALBERT, and DistilBERT, and to a lesser extent in RoBERTa.

- An approximation of subspace size could be determined by removing more dimensions. This would require more computational power (bigger GPUs) to handle increased batch sizes.

Both of the above observations hold across all four models. This means that subspaces exist in the high dimensional latent space of pre-trained models like BERT and its variants that encode information about biases. This result is interesting because our methodology does not assume any specific kind of bias and only uses toxicity as an example. The above methodology can be applied to obtain a subspace for any kind of harmful or unwanted bias. This confirms our original hypothesis that there exists a low dimensional bias subspace that encodes toxicity information.

6.2 Conclusions and Future Work

We conclude with the following remarks:

- The experiments and observations in previous sections lead us to believe that pre-trained language models encode bias in low dimensional subspaces. The bigger the subspace that was removed, the lesser the accuracy or higher the loss obtained.

- This method of de-biasing works without requiring fine-tuning of pre-trained models. Our methods can be applied to any bias, like gender, race, and religion, and researchers could use any existing classification dataset for these biases.

Our work touches upon an exciting branch of NLP research and has many potential areas to explore. We hope that this work will bring forth future work to understand and explore latent spaces of large models. Suggestions for future work include better construction of parallel data, including n-grams instead of single words. Also, non-linear methods of PCA such as Kernel PCA and other dimensionality reduction methods may learn better bias subspaces. Finally, analyses of other models, such as GPT2 and other generative models, could also be interesting.

References

Rishabh Bhardwaj, Navonil Majumder, and Soujanya Poria. 2020. Investigating gender bias in bert. *arXiv preprint arXiv:2009.05021*.

Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. 2016. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. *arXiv preprint arXiv:1607.06520*.

Sumanth Dathathri, Andrea Madotto, Janice Lan, Jane Hung, Eric Frank, Piero Molino, Jason Yosinski, and Rosanne Liu. 2019. Plug and play language models: A simple approach to controlled text generation. *arXiv preprint arXiv:1912.02164*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Javid Ebrahimi, Anyi Rao, Daniel Lowd, and Dejing Dou. 2017. Hotflip: White-box adversarial examples for text classification. *arXiv preprint arXiv:1712.06751*.

Samuel Gehman, Suchin Gururangan, Maarten Sap, Yejin Choi, and Noah A. Smith. 2020. Realtoxicityprompts: Evaluating neural toxic degeneration in language models. In *EMNLP*.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2019. Albert: A lite bert for self-supervised learning of language representations. *arXiv preprint arXiv:1909.11942*.

Paul Pu Liang, Irene Mengze Li, Emily Zheng, Yao Chong Lim, Ruslan Salakhutdinov, and Louis-Philippe Morency. 2020. Towards debiasing sentence representations. *arXiv preprint arXiv:2007.08100*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Paul Michel, Xian Li, Graham Neubig, and Juan Miguel Pino. 2019. On evaluation of adversarial perturbations for sequence-to-sequence models. *arXiv preprint arXiv:1903.06620*.

Reid Pryzant, Richard Diehl Martinez, Nathan Dass, Sadao Kurohashi, Dan Jurafsky, and Diyi Yang. 2020. Automatically neutralizing subjective bias in text. In *Proceedings of the aaai conference on artificial intelligence*, volume 34, pages 480–489.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9.

Emily Reif, Ann Yuan, Martin Wattenberg, Fernanda B Viegas, Andy Coenen, Adam Pearce, and Been Kim. 2019. Visualizing and measuring the geometry of bert. *Advances in Neural Information Processing Systems*, 32:8594–8603.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. 2019. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*.

Emily Sheng, Kai-Wei Chang, Premkumar Natarajan, and Nanyun Peng. 2020. Towards controllable biases in language generation. *arXiv preprint arXiv:2005.00268*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *arXiv preprint arXiv:1706.03762*.

Eric Wallace, Shi Feng, Nikhil Kandpal, Matt Gardner, and Sameer Singh. 2019. Universal adversarial triggers for attacking and analyzing nlp. *arXiv preprint arXiv:1908.07125*.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. *arXiv preprint arXiv:1906.08237*.