Understanding and Mitigating Unintended Demographic Bias in Machine Learning Systems

by

Christopher Sweeney

Submitted to the Department of Electrical Engineering and Computer Science
in partial fulfillment of the requirements for the degree of

Master of Engineering in Electrical Engineering and Computer Science

at the

MASSACHUSETTS INSTITUTE OF TECHNOLOGY

June 2019

Author .....
Department of Electrical Engineering and Computer Science
May 17, 2019

Certified by .....
Richard Fletcher
Research Scientist
Thesis Supervisor

Certified by .....
Maryam Najafian
Research Scientist
Thesis Supervisor

Accepted by .....
Katrina Lacurtz
Chairman, Department Committee on Graduate Theses

Understanding and Mitigating Unintended Demographic Bias in Machine Learning Systems

by

Christopher Sweeney

Submitted to the Department of Electrical Engineering and Computer Science
on May 17, 2019, in partial fulfillment of the
requirements for the degree of
Master of Engineering in Electrical Engineering and Computer Science

Abstract

Machine Learning is becoming more and more influential in our society. Algorithms that learn from data are streamlining tasks in domains like employment, banking, education, heath care, social media, etc. Unfortunately, machine learning models are very susceptible to unintended bias, resulting in unfair and discriminatory algorithms with the power to adversely impact society. This unintended bias is usually subtle, emanating from many different sources and taking on many forms. This thesis will focus on understanding how unfair biases with respect to various demographic groups show up in machine learning systems. Furthermore, we develop multiple techniques to mitigate unintended demographic bias at various stages of typical machine learning pipelines. Using Natural Language Processing as a framework, we show substantial improvements in fairness for standard machine learning systems, when using our bias mitigation techniques.

Thesis Supervisor: Richard Fletcher
Title: Research Scientist

Thesis Supervisor: Maryam Najafian
Title: Research Scientist

Acknowledgments

I would like to thank my research advisor, Maryam, for mentoring me through the research process. She was a constant source of motivation and encouragement throughout the ups and downs of this research project. I am fortunate to have worked with such a driven and talented researcher.

I would also like to thank my MEng supervisor, Richard, for providing an inviting and engaging community at MIT's DLAB. Thanks to his support, I had all the resources I needed to conduct research in fairness and machine learning.

This thesis was made possible in part through support of the United States Agency for International Development. The opinions expressed herein are those of the authors and do not necessarily reflect the views of the United States Agency for International Development or the US Government.

Contents

Chapter 1

Introduction

1.1 Machine Learning in Society

Machine Learning (ML) is having an increasingly prominent impact in society. Large companies are mining huge amounts of user data with little to no regulation on how the data is used, and governments are amassing large databases on the public to predict things like crime and recidivism rates. To combat the complexity and scale of the large sums of data in use, machine learning is often employed. Although powerful, ML algorithms are notoriously opaque, offering little insight into how predictions are being made on data. Additionally, because much of the harmful biases in society are reflected in data, machine learning algorithms often pick up on and exacerbate these biases. This creates a dangerous scenario where we are giving power to unpredictable algorithms, and even worse, feeding them data containing all the unfavorable aspects about society. With more decision making being handed over to machine learning algorithms, it is important for computer scientist to be aware of possible unintended biases encoded into the models they create.

Most of the work in understanding and preventing ML algorithms from making unfair decisions has been in academia. Unfortunately, the combination of complicated ethical questions and the diversity of possible ML applications has made it difficult for practitioners to transfer ideas from academia into industry. All the while, unfair ML algorithms already exist and are doing harm in society. For example, the talent

acquisition industry is being overhauled by algorithms learned from data that make decisions on who to hire. The influx of ML into this domain has been particularly quick in developing countries that have a large unemployed workforce with relatively few jobs. In scenarios like these, companies are turning to ML to sift through resumes or evaluate skill sets. Unfortunately, dangerous biases hinder the fairness of these technologies. For example, demographic groups with less employment data are at risk of more inaccurate predictions compared to groups with more data. Credit is another domain where the use of ML can lead to discrimination. Again, especially in the developing world where there do not exist established credit mechanisms, companies are resorting to ML algorithms that aim to predict the credit worthiness of an individual. A discriminatory algorithm can prevent access to fair loans for certain demographic groups. Another example domain lies in ML models for Natural Language Processing (NLP) that make predictions on text. As most text is written by humans, ML word corpora and training data reflect all the dangerous biases that exist in society. An algorithm trained to predict the sentiment of a customer review might correlate words from a particular demographic group (i.e Black, Hispanic) with negative concepts. It is important for researchers to understand the variety of situations in which dangerous biases can manifest, and develop solutions that are specific and interpretable. In this work, we prioritize specificity and interpretability, developing a variety of methods to understand and mitigate unintended demographic bias for applications in NLP.

1.2 Structure of Thesis

This thesis is organized as follows. In Section 2, we provide background for this thesis, defining demographic discrimination in the context of machine learning, and describing how people have thought about mitigating unintended bias in machine learning systems. In Section 3, we present our techniques for understanding and measuring unintended bias at various stages of the ML pipeline. Then in Section 4, we describe our techniques for mitigating unintended bias at the various stages of the

ML pipeline. In Section 5, we evaluate our mitigation techniques for typical machine learning systems in Natural Language Processing, showing improved fairness. Finally in Section 6, we conclude our work, recommending possible solutions for unsolved issues of fairness in machine learning.

Chapter 2

Background

We begin this chapter by providing a foundation for what it means for a machine learning algorithm to be unfair. Then, we describe the various ways researchers have defined fairness in the machine learning setting. Next, we explore how researchers have dealt with the various sources of unintended demographic bias that cause machine learning systems to be unfair. Finally, we transition our discussion to NLP, justifying our focus for applications in sentiment analysis and toxicity prediction.

2.1 Discrimination in Machine Learning

As ML algorithms often make decisions that directly impact people (i.e. predicting whether someone should receive a loan), they have the ability to perform in an unfair or discriminatory manor. Unfortunately, defining discrimination in the context of machine learning is difficult as it is already a complicated concept, legally speaking. To provide a richer base for understanding how machine learning algorithms can discriminate, it is useful to connect machine learning to legal frameworks of discrimination. In countries like the United States, there are many demographics such as age and race that are protected against discrimination by law. Generally, the law splits discrimination into two types: Disparate Treatment and Disparate Impact.

2.1.1 Disparate Treatment

Disparate Treatment describes a situation where a policy or decision is made that explicitly discriminates against a demographic group of people based on race, age, gender etc. In practical scenarios, machine learning algorithms are not trained to discriminate against certain demographics. However, due to the fairness blind optimization functions often used in practice, a machine learning algorithm may end up making decisions based on a protected attribute like gender. Unfortunately, the naive solution of removing all demographic attributes from the dataset does not work, as modern datasets tend to be large, containing many correlations between protected attributes and non protected attributes. This means ML algorithms can make discriminatory decisions even without explicitly considering protected attributes.

2.1.2 Disparate Impact

Disparate Impact is much subtler and more dangerous than Disparate Treatment. Disparate Impact involves policies or decisions that implicitly discriminate against a certain group of people. With the massive amounts of data we feed our machine learning algorithms, they are likely to find features that correlate with sensitive attributes such as race. Therefore even if an algorithm is not explicitly trained to make unequal decisions for different demographic groups, it may do just so. In other words, the discrimination is caused by *unintended bias*. Even worse, due to the complicated correlations between various features in a dataset, one cannot simply solve this issue by removing features pertaining to demographic attributes like race or gender. Furthermore, many realistic scenarios in machine learning involve data where protected attributes are not even available [8, 16]. This means we cannot define discrimination in machine learning based on the protected attributes used in training, but by the disparate decisions made at test time. This complicates the definition of fairness in machine learning because the decisions a ML algorithm makes has to be audited in many different scenarios. Researchers have tried to simplify the auditing of ML models by focusing the general framework of *group fairness*.

Group Fairness

Machine learning often creates a model that assigns a certain class or score to a person or attributes representing a person. Unfair differences in scores between different demographics groups is the type of inequality studied in *group fairness*. There are still many ways one can formally define fairness within this framework, but it is useful to evaluate inequality at the group level because it links nicely with legal concepts of discrimination by demographic group. We target this type of demographic discrimination in this work. There are many artifacts in the machine learning pipeline that can lead to a model being unfairly discriminatory. We refer to the various sources and causes of this unfairness as *unintended demographic bias*. Throughout this work, we describe attempts to both measure and mitigate unintended demographic bias in the effort to create machine learning algorithms that have a higher degree of *demographic fairness*.

There are also other regimes of measuring inequality in machine learning like *individual unfairness* [11], or more recent definition of fairness where demographic groups are determined from data [40, 2]. Though important to address, we limit our scope to understanding and mitigating unintended bias with respect to typical demographic groups (i.e race, gender, religion, etc.).

Demographic Fairness vs Demographic Bias

Within group fairness, there are many different aspects of machine learning predictions to consider before rating its fairness with respect to demographic groups. For example, the fairness of a ML binary classifier can be measured by the disparity between predictions for people in various demographic groups via true positive rate, negative rates, etc. To measure this *demographic fairness*, researchers have created many formal metrics that capture different aspects of inequality. All these metrics measure some sort of disproportionate favorable outcomes for people within different demographics. However, it is equally important to understand and measure *demographic bias*, which instead describes an artifact of the machine learning pipeline (i.e.

data, choice of algorithm, etc.) that leads to an unfair decision being made by the trained model. The term, *bias* is an overloaded concept in machine learning, but for this work, we focus on the types of bias in a ML system that cause unfairness. A more accurate and nuanced understanding of the unintended demographic bias in a ML system leads to more effective methods to mitigate the bias and create fairer algorithms. We discuss the various sources of demographic bias in more detail in section 2.3. In the next section, 2.2, we describe the various metrics of demographic fairness created in the academic community.

2.2 Measures of Demographic Fairness

We describe two genres of demographic fairness metrics. First we describe the more theoretical and formal notions of group fairness. Then we describe, more concrete measures of fairness for applications in NLP.

2.2.1 General Measures of Demographic Fairness

There are many definitions of fairness that have been created in the academic community to measure different aspects of inequality. Within our discussion on group fairness, there are two main definitions we will focus on: Demographic Parity and Equalized Odds [46, 15]. To describe these metrics, we imagine a toy scenario where we have a binary predictor, $\hat{Y}$, that predicts whether someone should receive a loan. The predictor is trained on data of the form, (X_i, Y_i) where $X_i \in \mathbb{R}^d$ is list of features describing a person and Y_i is the corresponding label where $Y_i = 0$ represents a defaulted loan and $Y_i = 1$ is a paid back loan. Furthermore, each X_i contains a sensitive attribute, $S \in \{M, F\}$ that is either male or female. The goal is to use X_i to predict if the given person would default on the loan. We define two formal notions of fairness for our predictor below.

- *Demographic parity* states that the decisions of a ML classifier should be prob-

abilistically independent of the protected attribute.

$$P[\hat{Y} = c|S = M] = P[\hat{Y} = c|S = F] \quad \forall c \in \{0, 1\}$$

This notion of fairness is simple, as it does not require knowing labels for the classification objective in question. However, many argue that this metric is not fair because it does not explicitly take into account the true label, Y.

- *Equalize odds* states that the decisions of a ML classifier should be probabilistically independent of protected attribute, given the true label, Y.

$$P[\hat{Y} = c|S = M, Y = y] = P[\hat{Y} = c|S = F, Y = y] \quad \forall y, c \in \{0, 1\}$$

This notion seems fairer as it explicitly conditions on the fact that someone was able to pay back a loan vs just the protected attribute. Unfortunately, this metric also is limited as one may not have enough labeled data to comprehensively characterize a demographic group.

Although these definitions have unified many of the attempts to make machine learning fairer, they are often too abstract and contrived for real world applications. Many constraints of implementing ML systems in practice make it difficult to prescribe an intuitive notion of fairness. For example, some of the definitions (equalized odds) implicitly require test time access to protected attributes to measure fairness, which is a strong limitation in practice. Furthermore, different types of data may require a more detailed fairness metric. For example in NLP, there does not exist an obvious definition of a protected attribute in text, making it difficult to define what a fair text classifier actually means. Natural Language Processing is an nice application focused area of ML where researchers have already realized the need for more task specific metrics of fairness.

2.2.2 Application Specific Measures of Demographic Fairness in NLP

Many of the more application specific measures of demographic fairness draw inspiration from the general metrics like equalized odds, but focus to the constraints of the problem at hand. This makes it much more tangible for practitioners to use in real world contexts. One area where meaningful metrics have been developed for more application specific purposes is in NLP. Figure 2-1 illustrates how general notions of fairness require more specification when applying them to a domain specific area like NLP.

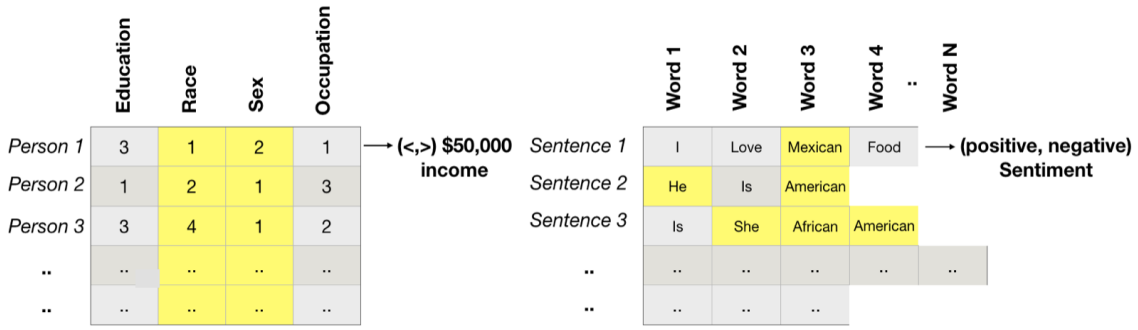


Figure 2-1: Framing fairness for an NLP based setting is more complicated than for non text-based settings. The left image portrays binary classification with respect to sensitive attributes (highlighted in yellow) of a data table from the Adult Dataset [32]. Sensitive attributes in sentiment classification (right image) could instead be words within a sentence. This complicates defining fairness as one has to define which words in the sentenced are sensitive attributes.

Many of the earlier mentioned fairness metrics rely on the ability to clearly define sensitive attributes. The left image in Figure 2-1 shows a tabular dataset similar to the well studied UCI Adult Dataset [32]. In this setting it is relatively easy to define fairness, where each row is a vector of features, where certain columns that describe sensitive attributes (highlighted yellow), should not explicitly be correlated with the binary predictions for whether the person makes greater or less than $50,000 a year. When evaluating fairness in an NLP setting like sentiment classification (right image in Figure 2-1), we can leverage many of the ideas from demographic fairness and equalized odds, but there needs to be more specification. In NLP, one can define

sensitive attributes not as columns, but words in a varying length vector. Each entry in the table is a n-dimensional word vector belonging to the row (sentence in this case). More complications could come from the protected attribute instead being the author of the sentence [13], but we focus on discrimination with respect to textual demographic protected attributes like identity terms (i.e. American, Jewish, Female, etc.). Researchers have used this framework to evaluate fairness in NLP applications such as *Sentiment Analysis* and *Toxicity Prediction*.

Two important problems in NLP are *Sentiment Analysis* and *Toxicity Prediction*. Although related, sentiment analysis deals with generally assessing the emotional content behind the text, and toxicity deals with the offensiveness or inappropriateness of written text. In sentiment analysis, words related to different demographic groups may receive unequal predictions for positive or negative sentiment. We call the cause of this unfairness, *Sentiment Bias*. In toxicity prediction, words related to different demographic groups may have an unequal chance as being flagged as toxic. We call the cause of this unfairness, *Toxicity Bias*. Researchers have defined application specific notions of fairness for each application.

To evaluate fairness in sentiment analysis systems, [24] created the Equity Evaluation Corpus (EEC), containing a large set of synthetic sentences designed to tease out unfairness in a ML model. Using the EEC as a benchmark dataset, [24] develops multiple fairness metrics. To evaluate fairness in toxicity prediction, [10] similarly uses a synthetic dataset to develop a fairness metric called Pinned AUC. In both works, the fairness metrics use demographic identity terms within a sentence to represent the sensitive attributes. In our experiments we create realistic NLP ML systems for applications in both *Sentiment Analysis* and *Toxicity Prediction* and evaluate fairness via the metrics described in these works.

2.3 Sources of Unintended Demographic Bias

There is a growing body of work devoted to finding unintended bias in machine learning. For example, researchers have found many different types of unfairness

[4, 18, 43] in NLP systems. Though it is important to find different types unfairness in the first place, in order to make ML models fairer, we need to understand where the unintended bias is coming from, and develop strategies to mitigate the bias. Since modern ML pipelines are very complicated, there are many sources from which unintended demographic bias can emanate. We discuss four main sources of harmful biases in ML systems: *Embeddings*, *Dataset*, *Learning Algorithm* and *Decision Level*, and attempts to measure and mitigate them. In the machine learning and fairness literature, bias mitigation techniques at the *Embeddings* and *Dataset* level are usually called *pre-processing* methods, techniques at the *Learning Algorithm* level are called *in-processing* methods, and techniques for the *Decision Level* are called *post-processing* methods.

2.3.1 Embeddings

In machine learning, embeddings are used to create computational representations of data. Many data points in the real world are categorical. The ability to represent these objects in a continuous vector space where distances between different objects encode a real meaning is very useful in ML. Applications of embeddings range from representing features in an image to representing words or sentences. Due to the widespread use of word embeddings in real world applications, we focus on methods to understand how unintended demographic bias shows up in word embeddings. Word embeddings can contain many different types of bias. [7] and [6] both found alarming biases in word embeddings with respect to race and gender. In order to more formally measure the unintended bias, [7] created the Word Embeddings Association Test (WEAT score). Although a good first step, the WEAT score has many limitations, and we develop a new metric to expand the versatility of measuring unintended word embedding bias.

Researchers have also found ways to mitigate unintended bias in word embeddings. [6, 49] attempt to prevent non-gendered words or phrases like *computer programmer* from being polarized towards one gender over another. [48] uses an adversarial learning technique to create fairer analogy completion tasks in word embeddings. We draw

inspiration from these techniques to mitigate unintended bias in sentiment analysis and toxicity prediction applications.

2.3.2 Dataset

The ML training set can also contain many types of unintended demographic bias. This bias is usually some sort of imbalance in the dataset. Imbalances can come from some demographic groups not having enough data, causing the ML algorithm to overfit to groups with more data. This can cause the ML algorithm to be less accurate towards groups with less data. Another type of dataset imbalance can come from unfair representations of demographic attributes. ProPublica found that COMPAS, a system that predicted recidivism rates was correlating higher recidivism with darker skin color [21]. This was a dataset imbalance where there was an unfair overrepresentation of dark skinned over light skinned people. Many researchers solve such dataset imbalances by directly augmenting the dataset with synthetic training data [10, 14, 35]. Others have tried to mitigate this bias through obfuscating demographic information in the dataset through either optimization [47, 22], or adversarial learning [3, 12]. Though a direct way to counteract bias, most of the above methods either add extra data to the training set, or find some way to hide data. As we do not have a perfect model for the real world, artificially changing the dataset could have unintended side effects.

2.3.3 Learning Algorithm

The choice of learning algorithm is a type of unintended bias that affects model fairness. Even different training runs of a model could result in large difference in model fairness. Since classical ML loss functions do not have fairness constraints, models are free to sacrifice fairness for accuracy. This problem is exacerbated by the natural overfitting that happens too often in machine learning, increasing training accuracy at the expense of real world performance and fairness. Though there are many ad-hoc methods to mitigate bias at the learning algorithm like trying out a

bunch of learning algorithms and evaluating model fairness, researchers have created more deliberate ways to add bias mitigation techniques into the learning algorithm. Works like [23] investigate methods to add fairness constrains to the loss function via regularization terms that directly minimize mutual information between the predictor and sensitive attributes. Other methods use the same objective but instead use the adversarial learning framework [48]. In addition to objective function modifications, researchers have also shown that careful handling of Kernel/PCA methods can create fairer ML algorithms [37]. Mitigating unintended bias at the learning algorithm is usually an effective technique, but it is often difficult to disentangle which source of bias is being mitigated (i.e embeddings, training set, learning algorithm). We develop mitigation techniques at the learning algorithm level and address this issue.

2.3.4 Decision Level

Unintended demographic bias may be introduced after a ML algorithm is trained through the choices of thresholds used on models like probabilistic classifiers. A demographic group-specific threshold, may introduce unintended bias by allowing a higher positive rate for one group over another. Since this bias is introduced at the very end of the ML pipeline, it is very closely related to the fairness of the ML systems as a whole. Modifying the post-training (prediction level) thresholds of a classification algorithm to comply with formal definitions of fairness is a very clear way to show that a ML algorithm will not discriminate. As pointed out by [15], *a model contains unintended bias if it performs better for some demographic groups than others.* [15] shows how one can calibrate demographic group-wise classification thresholds to meet various fairness metrics. However, as shown in [25, 9], in most cases, it is impossible to satisfy more than one group fairness metric at a time. This makes it very challenging to mitigate unintended bias solely at the decision level.

2.4 NLP as a Focus

Many of the previous mentioned works offer great insights into dealing with unintended bias in various applications. However, applications of ML span many data types (i.e images, features, text, etc) and prediction types (classification, regression, clustering etc.). This requires more grounded and application specific metrics and mitigation techniques for unintended demographic bias. We have already shown the breadth of work focusing on fairness in machine learning systems for NLP. NLP is a useful area to study for two reasons. Firstly, NLP adds many additional challenges for machine learning pipelines like biased word embeddings and less structured training data. Secondly, NLP data is widely used and accessible. In other applications that deal with tabular data, like banking or healthcare, there are very few datasets available for researchers to test their bias mitigation techniques. This is often due to the unwillingness of corporations to release possibly personal information nor receive bad press from dangerous bias discovered in the released datasets. In NLP on the other hand, the rise of social media has made many text datasets related to sentiment analysis widely used and available. For this reason, we use NLP as a focus for investigating demographic bias in machine learning systems. More specifically, we use target applications in text sentiment analysis and toxicity prediction where our goal is to mitigate unintended demographic bias. For the rest of this work, we leverage much of the existing work in fairness for machine learning to develop more specific tools for NLP applications.

2.5 Summary

In this chapter, we defined our problem setting. We first connected legal definitions of fairness like disparate impact to ML, then centered in on the concept of group fairness, highlighting useful demographic fairness metrics for applications in NLP. We also went through each stage in a typical machine learning pipeline, and covered how each step can introduce unintended demographic bias into a ML system. Ad-

ditionally, we covered a wide array of techniques researchers have used to measure and mitigate unintended demographic bias at different stages of the ML pipeline. In this thesis, we focus on mitigating and understanding unintended demographic bias at the embedding and learning algorithm level for applications in sentiment analysis and toxicity prediction.

Chapter 3

Understanding and Evaluating Unintended Demographic Bias

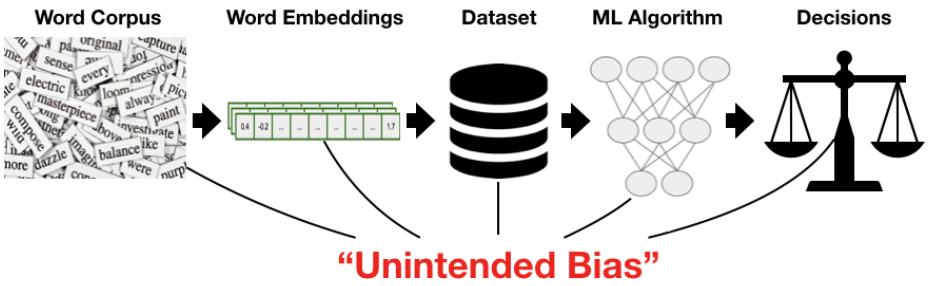


Figure 3-1: Typical machine learning system for NLP applications. Unintended Bias can enter at any stage of the system. It is important to understand and mitigate unintended bias at all levels of the ML pipeline.

There are many works geared at understanding and evaluating how machine learning algorithms produce unfair results. One of the big takeaways from research in this field is that different tasks need application specific measures of fairness. Due to the nuanced concept of fairness in legal and social settings, it is often difficult to create general measures of fairness. Even more concerning, the unfairness in a ML algorithms could be caused by a number of forms and sources of bias. Figure 3-1 shows a typical pipeline for ML applications in NLP. Unintended bias can enter at any stage due to human choices made at each step. The ability to identify and measure bias coming from various stages of the ML pipeline would be largely beneficial to prac-

titioners. To measure unintended demographic bias, we define more specific metrics with respect to sentiment analysis and toxicity prediction applications. Furthermore, we focus on measuring this bias at the word embedding and learning algorithm stage.

3.1 Word Embeddings

Word embeddings have established themselves as an integral part of Natural Language Processing (NLP) applications. Recent studies have shown that word embeddings exhibit unintended gender and stereotype biases inherent in the training corpus. For instance, if the word pairs frequently co-occur in the word corpus training set, then their word embeddings also tend to be more similar, and can encode gender and racial bias. Examples of such word pairs can be found in the word embeddings trained on Google News data: "he-janitor", "she-housekeeper","he-alcoholism", "she-eating disorder", "black male-assaulted", and "white male-entitled" [5].

Bias can be defined as an unfair expression of prejudice for or against a person, a group, or an idea. Bias is a broad term, which covers a range of problems particularly relevant in natural language systems such as, discriminatory gender bias [5, 49], bias against regionally accented speech [31, 30], personal or political view bias [20, 38], and many other examples. In this work, we restrict our definition of unintended bias to unequal distributions of negative sentiment among protected groups in text. One could also look at unequal distributions of positive sentiment or toxicity, but we present our techniques for the negative case.

3.1.1 Lack of Interpretability in Word Embeddings

As the number of people commenting on social media platforms increases, moderating the large amount of text becomes difficult [17]. For this reason, using NLP for automatically detecting abusive language or negative sentiment is of high importance. Even more important, preventing negative sentiment from mixing with sensitive attributes (i.e. race, gender, religion) in word embeddings is needed to prevent discrimination in ML models using the embeddings. As studied in [33], unintentionally

biased word embeddings can have adverse consequences when deployed in applications, such as movie sentiment analyzers or messaging apps. Negative sentiment can

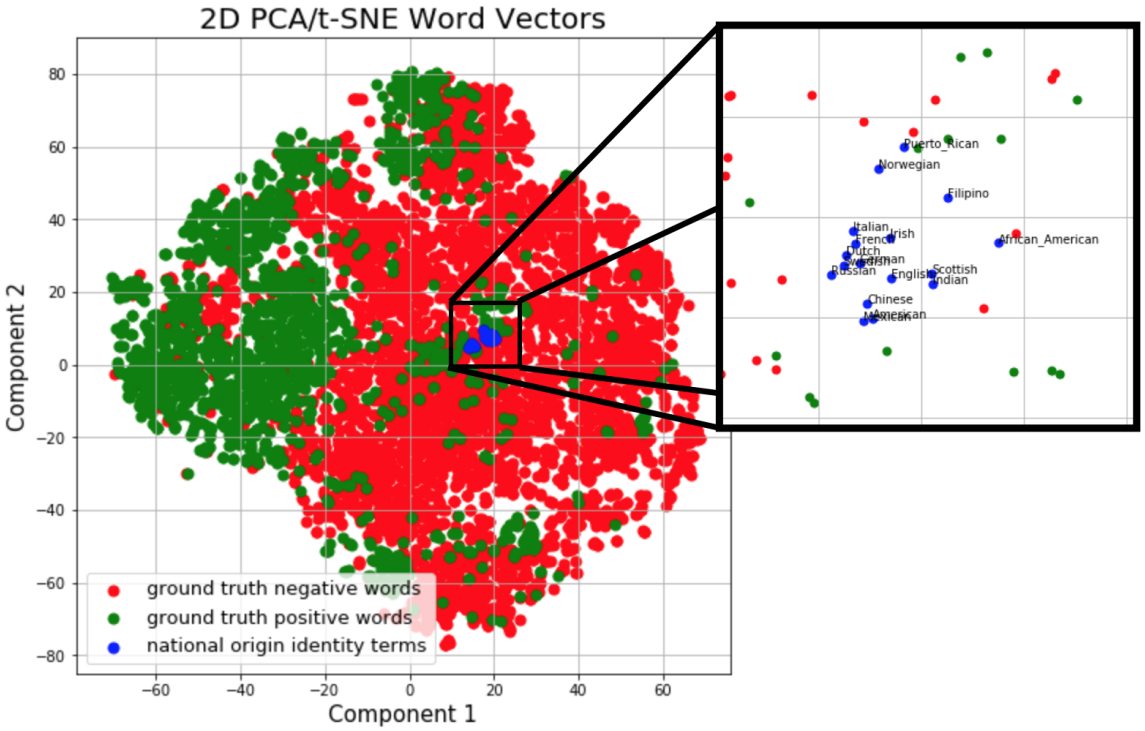


Figure 3-2: 2-D PCA embeddings for positive/negative sentiment words and a set of national origin demographic identity terms. Geometrically, it is difficult to parse how these embeddings can lead to discrimination.

be unfairly entangled in the word embeddings, and detecting this unintended bias is a difficult problem. We need clear signals to evaluate which groups are discriminated against due to the bias in a word embedding. That way we can specifically pinpoint where to mitigate those biases. To demonstrate this need for clear signals of fairness in word embeddings, we look at Figure 3-2. Figure 3-2 shows a 2D word embedding projection of positive (green) and negative (red) words. It would be unfair for any given national identity word vector (blue) to be more semantically related to negative terms than the other identities. However, many identity terms exists closer to negative words than other identity terms in the vector space. This hints there might be an unintended bias, but the vector space has no absolute interpretable meaning, especially when it comes to whether this word embedding will lead to a unfairly discriminative ML algorithm. We present a framework that enables transparent insights

into word embedding bias by instead viewing the output of a simple logistic regression algorithm trained on an *unbiased* positive/negative word sentiment dataset initialized with *biased* word vectors. We use this framework to create a clear metric for unfairness in word embeddings.

Modifying the post-training (prediction level) thresholds of a classification algorithm to comply with formal definitions of fairness is a very clear way to show that a ML algorithm will not discriminate. As pointed out by [15], *a model contains unintended bias if it performs better for some demographic groups than others.* This is a clear definition that is lost in unintended bias analysis for word embeddings. Many tests proposed to evaluate word embedding bias leverage vector space arguments between only two identities at a time like *man* vs *woman* [5], or *European African names* vs. *African American names* [7]. Though geometrically intuitive, these tests do not have a direct relation to discrimination in general. We present a framework and accompanying metric, Relative Negative Sentiment Bias (RNSB), to enable a clear evaluation of discrimination with respect to word embedding bias for any protected group.

3.1.2 Relative Negative Sentiment Bias

We present our framework for understanding and evaluating unintentional demographic bias in word embeddings. We first describe the flow of our framework. Then, we show how our framework can enable insightful analysis and new fairness metrics like Relative Negative Sentiment Bias (RNSB).

Our framework enables the evaluation of unintended bias in word embeddings through the results of negative sentiment predictions. Our framework has a simple layout. Figure 3-3 shows the flow of our system. We first use the embedding model we are trying to evaluate to initialize vectors for an unbiased positive/negative word sentiment dataset. Using this dataset, we train a logistic classification algorithm to predict the probability of any word being a negative sentiment word. After training, we take a set of neutral identity terms from a protected group (i.e. national origin) and predict the probability of negative sentiment for each word in the set. Neutral identity

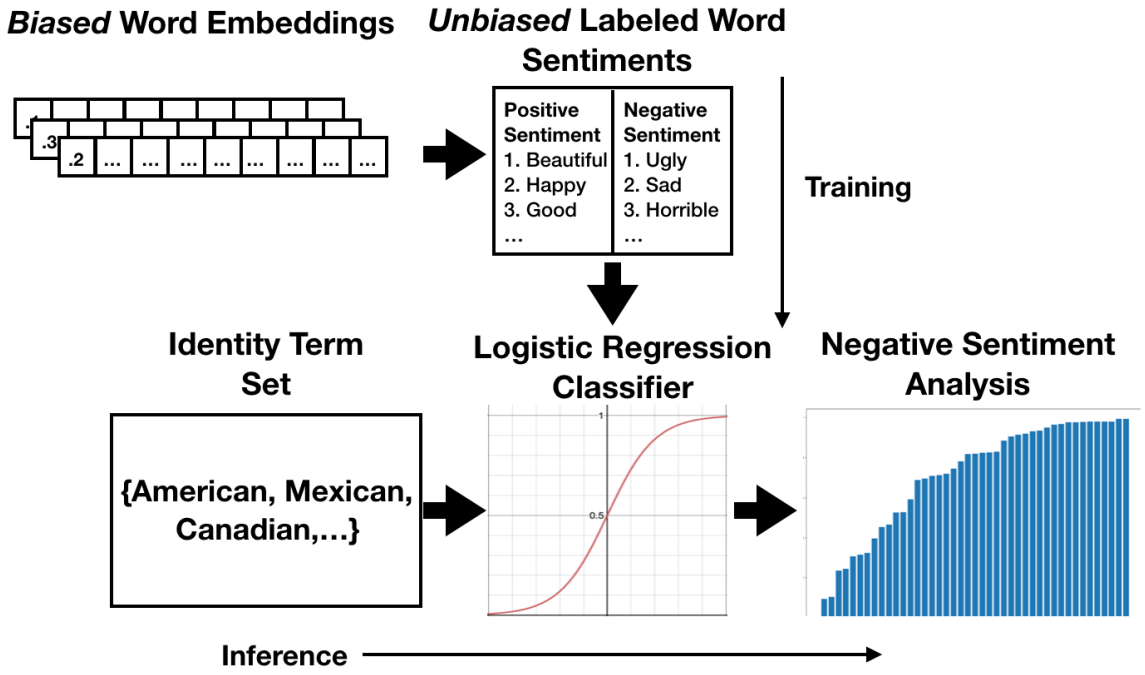


Figure 3-3: We isolate unintended bias to the word embeddings by training a logistic regression classifier on a *unbiased* positive/negative word sentiment dataset (initialized with the biased word embeddings). We measure word embedding bias by analyzing the predicted probability of negative sentiment for identity terms.

terms that are unfairly entangled with negative words in the word embeddings will have a higher likelihood of being predicted as negative sentiment. We use this set of negative sentiment probabilities to analyze the relative unfairness between identities within a protected group.

We now present our framework and metric for fairness, RNSB. For a set of gold standard labeled positive/negative semantic words (x_i, y_i) in training set, S, where x_i is a word vector from a possibly biased word embedding model, we find the minimizer, $f^*(x_i) = \sigma(w^T x_i)$, for the logistic loss, l, and learned weights, w.

$$min_{w \in R^d} \sum_{i=0}^{n} l(y_i, w^T x_i) + \lambda \|w\|^2, \lambda > 0$$

Then for a set, $K = \{k_1, ..., k_t\}$, of t identity keyword word vectors from a particular protected group (i.e. national origin, religion, etc.), we define a set, P, containing the predicted negative sentiment probability via minimizer, f^*, normalized to be one

probability mass.

$$P = \left\{ \frac{f^*(k_1)}{\sum_{i=1}^{t} f^*(k_i)}, ..., \frac{f^*(k_t)}{\sum_{i=1}^{t} f^*(k_i)} \right\}$$

Thus, our fairness metric, $RNSB(P)$, is defined as the KL divergence of P from U, where U is the uniform distribution for t elements.

$$RNSB(P) = D_{KL}\left(P \| U\right)$$

We choose our set of neutral identity terms based on the most populous demographics for each protected group. However, due to the simplicity of this method, one can easily adapt it to include identity terms that suite the application in need of unintended bias analysis.

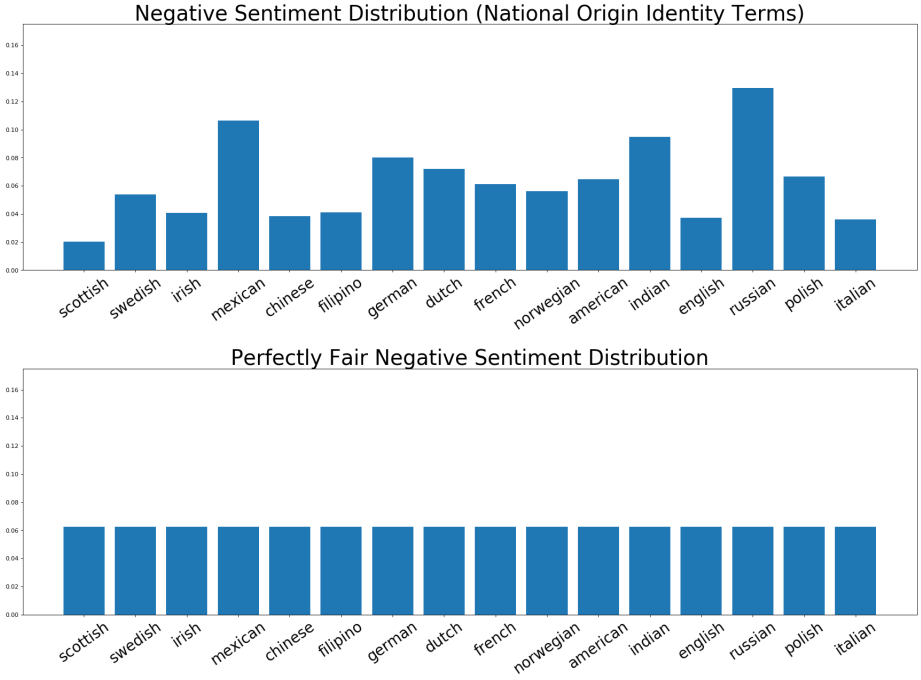


Figure 3-4: The top figure shows a distribution of negative sentiment predictions for an unfair word embedding model. Ideally, there would be no difference in negative sentiment predictions for different identity terms (bottom figure). RNSB measures unintended demographic bias via distance between these two distributions in a probabilistic sense using KL divergence.

Since neutral identity terms are inherently not associated with a sentiment, it is unfair to have identity terms with differing levels of negative sentiment. This type

of discrimination can show up in many downstream sentiment analysis applications. Thus, we want to eliminate differences between negative sentiment predictions of various identity terms. Mathematically, the fairest scenario can be represented as a uniform distribution of negative sentiment probability for identity terms from a protected group. Figure 3-4 shows an unfair and fair distribution of negative sentiment between demographic identity terms. Our RNSB metric captures the distance, via KL divergence, between the current distribution of negative sentiment and the fair uniform distribution. So the fairer the word embedding model with respect to sentiment bias, the lower the RNSB metric. Finally, because we use a gold standard set of positive and negative sentiment words along with a simple logistic regression model, we minimize the possibility that bias from the dataset or learning algorithm enters the system. This allows us to make stronger claims about the unintended bias contained to the word embeddings.

3.2 Learning Algorithm

Measuring unintended demographic bias at the learning algorithm level is difficult because the bias seen in a model's predictions could be caused by a complex mixture of word embedding, dataset and learning algorithm unintended bias. For example, a word embedding might introduce an unfair initialization of a word vector, a training set may have imbalanced data, and a certain choice of learning algorithm may converge to make more unfair predictions. In the previous section, to measure word embedding bias with RNSB, we used an unbiased dataset and simple logistic regression model to make sure we were not compounding biases at various stages in the ML pipeline. We evaluated bias via analyzing the distribution of negative sentiment predictions for various demographic identity terms. We can use the same concept to evaluate bias at the learning algorithm prediction level. However, with real world datasets and more complex models, we must keep in mind that measuring bias at the learning algorithm level will be more convoluted.

To truly sift out the source of unintended bias, one often has to perform the more

manual task of keeping upstream items like embeddings and training set constant, while changing the learning algorithm and measuring some change in the model's predictions.

3.2.1 Prediction Level Auditing

We define a metric specific to negative sentiment in text to audit unintended bias at the learning algorithm level. The intuition behind our metric is that inherently neutral identity terms belonging to a protected group should not have uneven chances for being classified as negative sentiment. This metric is very related to RNSB, but rather than manifesting biases in word embeddings, we measure the distribution of negative sentiment among demographic identity terms for a real world predictor. We call this metric Prediction Level Demographic Bias (PLDB). It is important to note that PLDB can also measure demographic bias with respect to any concept in text for which we want each demographic to have an equal prediction value. For example, in the next section, we use PLDB with respect to toxicity to evaluate mitigation techniques at the learning algorithm level. We now formalize our PLDB metric.

For a set, $\hat{P}$, of negative sentiment probability predictions from our predictor $f^*(x)$ for identity term word vectors within a protected group (ex. Mexican, American), we define the fairness of $f^*(x)$ with respect to a certain protected group (i.e. race, religion, etc.) as the KL Divergence between distribution P, (set $\hat{P}$ normalized to be a probability distribution) and the uniform distribution.

$$r_j = D_{KL}(P||U) \tag{3.1}$$

Where r_j is the PLDB metric with respect to one protected group (i.e. race, religion, gender).

Ideally, different identity terms would receive equal treatment, or in our case, equal negative sentiment. The perfectly fair scenario by our definition is the uniform distribution over a set of identities within a demographic. Our metric measures negative sentiment bias for a classifier as the KL divergence between the top and bottom

distributions. The PLDB metric also has a direct relation to mutual information between the predictor $f^*(x)$ and the set of sensitive attributes (identity terms), P. Fairness measures using mutual information have been used before in non text-based settings like in [23]. A uniform distribution among negative sentiment for identity terms would yield no amount of information about the predictor, which is a mutual information of 0.

3.3 Summary

In this section we presented two approaches to measure unintended demographic bias in ML for NLP systems. We focused on measuring bias at two levels, the word embedding level, and the learning algorithm level. We presented two metrics, RNSB for word embeddings and PLDB for the learning algorithm. Although both metrics are presented in the context of measuring negative sentiment bias, they are adaptable to other domains in NLP. In Section 5, we use the metrics to measure unintended demographic bias in real word NLP systems.

Chapter 4

Mitigating Unintended Demographic Bias

Once we can measure and understand bias at various stages in the ML pipeline, we can more effectively mitigate the unfair biases at their source. Researchers have shown promising results in mitigating unintended demographic bias in many aspects of machine learning. For this work, we develop techniques to mitigate unfair biases at the *word embeddings* and the *learning algorithm* level. We believe these two levels to be the most practical, introducing fewer unintended side effects compared to modifying the training set or decision thresholds of a ML algorithm. Furthermore, we focus on biases in the context of sentiment analysis and toxicity prediction for text. In our experiments, we compare how our methods compare to other debiasing techniques in terms of the overall accuracy and fairness of the ML algorithm.

4.1 Word Embeddings

The creation of word embeddings through algorithms like word2vec [27] led to increased performance in NLP tasks like sentiment analysis and toxicity prediction but also served as a new vector for unintended bias. Recently, works like [6] and [7] have shown that word embeddings can contain many types of bias related to gender and race. For our context, we focus on word embeddings that contain unintended

bias due to neutral demographic attribute word vectors having unfair polarization towards positive or negative sentiment, or unfair polarization towards toxicity. We define a demographic attribute term as a one word description used to assign a person to a particular demographic of a protected group. This can range anywhere from national origin terms like *American, Mexican*, religious identifiers like *Catholic, Jewish*, gendered words such as *female, male*, or even names that tend to belong to African American demographics like *Darnell, Lakisha*. The unequal treatment of demographics via such textual demographic attributes is very concerning given how entangled sentiment analysis and toxicity prediction are with domains that directly impact society. A downstream sentence toxicity classifier could pick up on this unintended word embedding bias and flag a sentence as toxic or abusive simply due the presence of protected attributes like demographic identity terms. In a perfectly fair scenario, identity terms describing inclusion in a certain demographic should be neutral with respect to sentiment and toxicity. In other words, if one substitutes the identity term in a sentence out for other identity terms, you should not see differing predicted results. We refer to this unfair distribution of sentiment (positive or negative) or toxicity among identity terms as sentiment bias and toxicity bias, respectively. Unfortunately unlike gender bias, which has been thoroughly explored in the research community, sentiment bias is a fairly loose concept. Consequently it is a much more difficult problem to decorrelate word vectors with sentiment, while retaining the usefulness of the embeddings. However, adversarial learning techniques have recently proven to be a powerful tool for reducing unintended bias in machine learning [48, 12, 3, 39]. We show how we can reduce both sentiment bias and toxicity at the word embedding level, using adversarial learning to decorrelate demographic identity term word vectors with toxicity and sentiment polarity, thereby, removing much of the unintended demographic bias at its source. We present our adversarial learning for removing sentiment polarity and then describe how we use a similar technique to remove toxicity polarity.

4.1.1 Using Adversarial Learning to Decorrelate Protected Attribute Word Vectors with Sentiment

We present an adversarial learning method to decorrelate word vectors with sentiment bias. Our algorithm is trained on a large corpus of words, but only identity terms are re-embedded with the model's debiased predictions. In presenting our approach, we first formally describe sentiment polarity in terms of vector subspaces. Then we present our adversarial learning algorithm that debiases word vectors with respect to these subspaces.

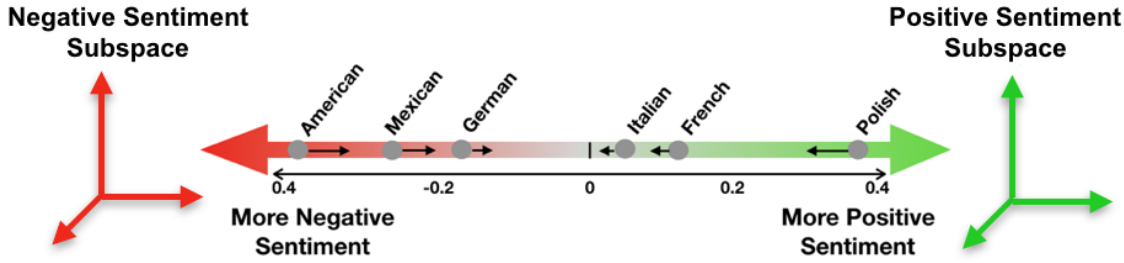


Figure 4-1: Demographic identity terms naturally have different sentiment polarity within an embedding model. Our goal is re-embed identity term word vectors with as little sentiment polarity as possible. By decorrelating word vectors with sentiment direction, we can move all identity terms towards a neutral point with no projection onto sentiment direction.

Creating the Sentiment Polarity Subspace

Since word embeddings are not explicitly labeled with dimensions that correspond to sentiment, we must define subspaces that capture negative or positive sentiment and assess word vector sentiment polarity by some distance metric to each subspace. [6] and [48] use a similar method to investigate gender bias that pairs gendered words like male and female, and extracts a directional vector from the PCA decomposition of a set of these pairs. We tried this approach, but unfortunately it does not work well for sentiment as the semantics for what defines a positive or negative concept is much looser than for gender. Additionally corresponding positive and negative words do not have as clear pairwise mappings as gendered words (ex. male:female).

To mitigate these two issues, we first take the most significant PCA component of a matrix of positive word vectors, and do the same with negative word vectors from the Sentiment Lexicon dataset [19]. We then take the signed difference between these positive and negative principal components. We found that the first significant component contains significantly more variance than any other component. We call the resultant vector our directional sentiment vector, as it connects the positive and negative subspaces. Given a word vector, we can now project it onto the directional sentiment vector to assess its sentiment polarity. A visualization of the resulting sentiment polarity for some national origin identity terms is displayed in Figure 4-1.

For our experiments, more positive the projection, the more positive sentiment polarity, and more negative corresponds to more negative sentiment polarity. The exact reasons identity terms have differing sentiment is not clear, but the effect of differing sentiment in downstream algorithms can be unfair. We choose to equalize the sentiment between identity terms by decorrelating the word vectors with sentiment, pushing the identity terms closer to 0, or in vector terminology, minimizing the projection onto the directional sentiment vector.

Validating the Directional Sentiment Vector

We need to validate that the directional sentiment vector effectively captures sentiment polarity. Therefore, we measure the accuracy of the following classification model.

$$\hat{y^*} = sign(k^t x_i^*)$$

where (y^*, x^*) is a labeled sentiment pair from the Sentiment Lexicon dataset [19]. The resulting precision of this classifier on the lexicon was 91.3%. This assures that the directional sentiment vector, k, contains much of the necessary information to determine sentiment polarity.

The Adversarial Learning Algorithm

We use an adversarial training regime to subtract out the unwanted sentiment correlations from our word vector. We define our depolarized sentiment vector, $\hat{y}$, as $\hat{y} = y - ww^T y$, for some learned weights w, and possibly sentiment biased word vector, y. To train and find weights, w, we define two competing objectives. Firstly, we want to make sure that the adversarial objective does not distort the meaning of the word vector. Since the word embedding vector space is not very interperatable, there does not exist an obvious loss function that encodes word vector distortion. Therefore we describe a simpler loss function, L_p, that minimizes the mean squared distance between our input word vector and debiased word vector, $(y - \hat{y})^2$. Conversely, for our adversarial objective, we want to pull y away from the positive and negative sentiment subspaces. We define an adversarial objective, L_a describing the ability for an adversary to predict the polarity of the word vector. As described in the previous section, sentiment polarity is defined by the projection of a word vector onto the directional sentiment vector, k. The adversary therefore tries to predict the sentiment polarity, z from the input word vector, $z = k^t y$. The adversary's prediction is defined as, $\hat{z} = w_a^T \hat{y}$, for adversarial weights w_a, learned with mean squared distance loss, $(z - \hat{z})^2$, L_a. An illustration of our adversarial model is shown in 4-2. We combine L_a and L_p using the methodology described in [48]. We minimize the following objective.

$$\nabla W L_{p_w} + proj_{\nabla W L_a} \nabla W L_p - \alpha \nabla W L_{a_w}$$

The middle projection term ensures that L_p does not end up helping our adversary.

Our algorithm learns how to take sentiment polarity our of any word vector by minimizing an adversary's ability to predict a word vector's sentiment polarity. We train our algorithm on a large corpus of words so it can generalize and learn to reduce sentiment polarity for any word vector. This is especially useful if a practitioner needs to pick and choose terms to debias that have not necessarily been in our training set.

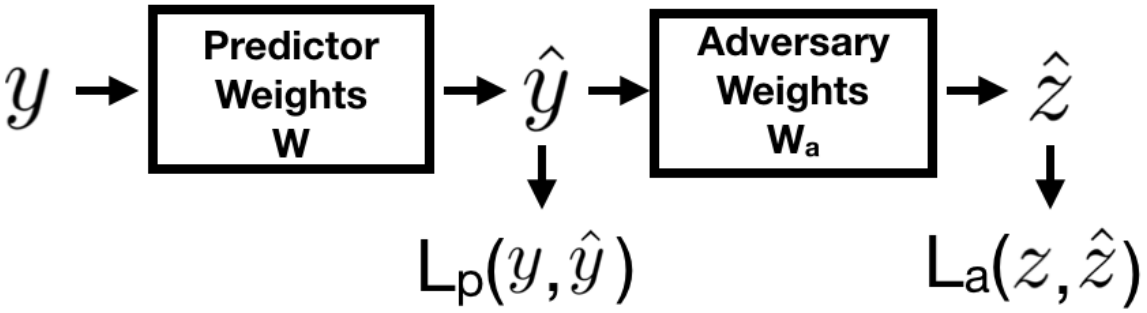


Figure 4-2: Our adversarial model architecture for debiasing word embeddings.

4.1.2 Using Adversarial Learning to Decorrelate Protected Attribute Word Vectors with Toxicity

Extracting meaning and intent from written text is the central goal in many NLP applications. We can use the same methodology described in the last section to depolarize word vectors with other concepts. Apart from sentiment, there are other aspects of language like toxicity, that are important to infer in real world applications. Toxicity bias is much like sentiment bias: it would be unfair for different protected attributes to have different correlations with toxicity. To adapt our method to decorrelate word vectors with toxicity, all we have to do is construct new subspaces in the word embeddings that capture non toxic and toxic concepts. In other words we have to create a directional vector connecting the two subspaces and then train the adversarial algorithm in the same way. Unfortunately, creating a directional vector connecting nontoxic and toxic concepts is more difficult as a word's toxicity is very related to the context in which the word is used. In the next section, we address how we overcome this challenge by searching for the right words to represent toxic and nontoxic subspaces.

Finding Representative Words for Toxic and Nontoxic Sentiment

Even though toxicity is a hard concept to pin down, certain positive and negative sentiment words can serve as good substitutes to represent nontoxic and toxic subspaces. We need to find these words. A good directional toxicity vector between these sub-

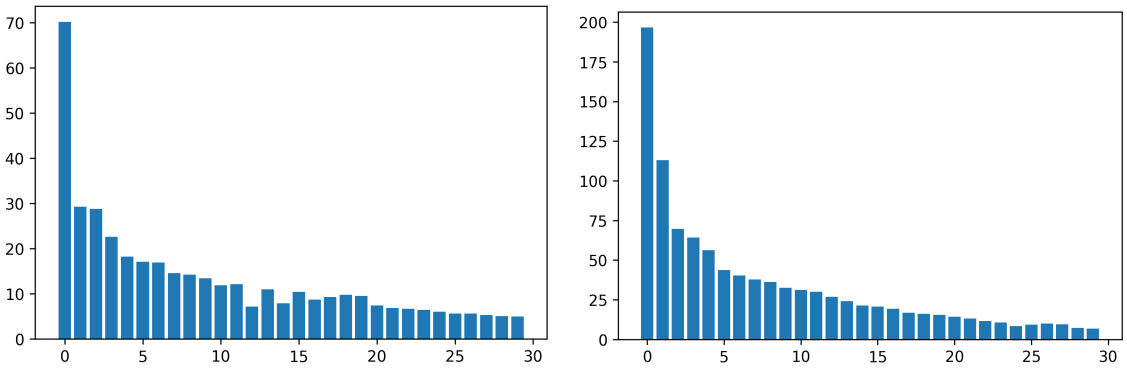


Figure 4-3: Top 30 principal components of the nontoxic word matrix (*left*), and toxic word matrix (*right*). In both cases, the first principal component contains much of the signal for representing toxic and nontoxic sentiment in the embedding vector space.

spaces, k, created from a set of positive and negative sentiment word vectors, should capture much of the same information as a good toxicity classification algorithm (we describe how exactly k is created from the set of words in the next section). That way, we know our toxic and nontoxic subspaces are good representations for what parts of the vector space are actually toxic. To this end, we evaluate the agreement of a Convolutional Neural Net based toxicity classification algorithm from [10] with a AUC of .968 on the Wikipedia Talk pages toxicity test set [45] with the classification algorithm presented below.

$$\hat{y^*} = sign(k^t x_i)$$

Where x_i is a positive or negative sentiment word vector from the Sentiment Lexicon dataset [19], and $\hat{y^*}$ is the prediction of toxicity (-1 for toxic and 1 for nontoxic). We choose 100 random samples from the Sentiment Lexicon dataset, each containing 50 unique positive and negative sentiment words. We created the vector, k, from each of theses samples. From the 100 samples, we chose the vector, k, that led to the highest agreement between $\hat{y^*}$ and the toxicity classification algorithm. The highest agreement was 96.3%. This assures that k contains the necessary information to determine whether a given word is more toxic or more nontoxic. Next, we dive deeper into how k is actually created from the most appropriate 50 positive and negative sentiment word vectors.

Constructing a Directional Vector between Toxic and Nontoxic Subspaces

To create the directional toxicity vector, k, we perform PCA on a matrix of the 50 positive sentiment word vectors, and on a matrix of the 50 negative sentiment word vectors. Figure 4-3 shows the first 30 dimensions for the negative sentiment and positive sentiment word vector matrices. We found that the first significant component contains significantly more variance than any other component. Therefore, we then take the signed difference between the first component of the positive sentiment and negative sentiment principal components, and call the resulting vector our directional toxicity vector.

Given a word vector, we can now project it onto the directional toxicity vector to assess its toxicity polarity. Our goal is to minimize the projection onto the directional toxicity vector for demographic identity terms, thereby decorrelating the vector with toxicity. We accomplish this goal using adversarial learning, and show that we can create fairer toxicity classification algorithms after debiasing the word embeddings.

4.2 Learning Algorithm

There are many ways to modify the learning algorithm to mitigate unintended demographic bias. Even the variance in multiple training runs for a model or the choice of model itself can have a huge impact on the resulting model's fairness. We develop methods to more directly mitigate unintended demographic bias at the level of the learning algorithm using PCA and regression. For the purposes of presentation, we focus our techniques on the task of toxicity prediction in text, but it can be applied to a wider variety of NLP applications like sentiment analysis. In presenting our mitigation techniques, we first describe our problem formulation for toxicity classification. Next, we discuss how we use our definition of unintended bias (PLDB) as a regularization term in toxicity classification. We call our regularization term Concept Regularization (CR). Finally, we explore how we mitigate unintended bias by removing principal components with higher PLDB measures. We call this method Leave-One-Out-PCA (LOOPCA).

Problem Setup

We focus on toxicity classification trained on a dataset, S, of embedded sentences labeled as toxic or non toxic (-1 or 1 respectively).

$$(x_i, y_i) \in S, \ \ \|S\| = n \ \ y_i \in \{1, -1\}$$

A popular baseline technique for incorporating multiple words of a sentence into a single sentence vector is to average them. So for a predictor $f^*(x)$, trained on dataset, S, we predict the probability of toxcicity as the average of the word vectors in a sentence that are not stop words from set, G (i.e. the, is, at, etc.).

$$x_i = \frac{x_i^1 + ... + x_i^k}{k}, \ \ \forall x_i^j \notin G$$

$$f^*(x) = P(x = -1)$$

4.2.1 Concept Regularization

Traditionally, regularization terms are added to a optimization problem to prevent overfitting. In our scenario we are trying to prevent our classification algorithm from fitting to solutions that contain more unintended bias. We leverage our PLDB metric to create a regularization term in a classification algorithm. We call our regularization term Concept Regularization or CR. CR can be applied to any scenario where one would like predicted concepts from text to be equalized between various demographic groups. In this presentation, we focus on equalizing disparate toxicity predictions for different demographic groups. We formally describe logistic regression with CR below.

We find the Expected Risk Minimizer (ERM), $f^*(x)$, for the logistic loss, l, with k CR terms, r_j, for $0 < j \leq k$, corresponding to a certain protected group (ex. $r_1 \rightarrow$ race, $r_2 \rightarrow$ religion, $r_3 \rightarrow$ gender, etc.) Each r_j is structurally the same as our PLDB

metric, now used to minimize toxicity bias for a particular protected group.

$$f^*(x_i) = \sigma(w^T x_i)$$

$$\min_{w \in R^d} \sum_{i=0}^{n} l(y_i, w^T x_i) + \lambda \|w\|^2 + \beta_1 r_1 + ... + \beta_k r_k$$

Where $\lambda, \beta_j \geq 0$. β_j are important tuning nobs that can help us arrive at desired solutions for different domains. For example, in some cases it might be more important to regularize the algorithm with respect to one protected group by increasing its corresponding weight term, β_j. Each regularization term r_j is the same as our PLDB, namely, the KL divergence between normalized predicted probabilities of a set, P, of d identity term word vectors in a protected group (ex. Christian, Jewish, etc. for the protected group of Religion) and the uniform distribution. For simplicity we define $\sigma(z) = \frac{1}{1+e^{-z}}$ and $z = w^T p_h$ for the h^{th} word vector in the set of identity term word vectors, P.

$$r_j = D_{KL}(P||U) = \sum_{h=1}^{d} \frac{\sigma(z)}{T} log\Big(\frac{\sigma(z)d}{T}\Big)$$

$$T = \sum_{h=1}^{d} \sigma(z)$$

Taking the gradient of each regularization term and the loss function, we arrive at the following gradient descent update.

$w_{t+1} =$

$$w_t - \eta\Big(\sum_{i=1}^{n} -y_i x_i \sigma(-y_i w_t^T x_i) + \lambda 2 w_t + \nabla R\Big)$$

$$\nabla R = \sum_{j=1}^{k} \beta_j \nabla r_j$$

$$\nabla r_j = \sum_{h=1}^{d} \frac{\sigma(z)(1-\sigma(z))}{T} (log\Big(\frac{\sigma(z)d}{T}\Big) + 1)$$

By minimizing the differences in toxicity predictions among identity terms within a protected group, we minimize the ability for different identity terms to deferentially add toxicity to a sentence. Our approach also generalizes to identity terms that

are not included in the regularization terms due to the proximity of related terms in the word embeddings. For example, the identity term, *Irish*, might exist in our regularization term, but since *Irish* is geometrically close in vector space to other protected attributes in the demographic (i.e. Celtic), our regularization term also minimizes the relative impact similar word vectors have on toxicity.

4.2.2 Leave-One-Out-PCA

Adding a regularization term to an optimization problem typically reduces overfitting to random noise. It is also known that that removing the insignificant singular vectors of the data matrix, which are essentially the random noise within the data, acts equivalently to regularization. This is known as principal component regression. Intuitively, it would follow that one could remove components that overfit to representations that correlate with large PLDB.

In our problem setup, the data we train on is a $n \times m$ data matrix, where n is the number of sentences in our dataset, and m is the number of dimensions in the word embeddings we use. Unfortunately because our data matrix is mix of averaged word embeddings for a dataset of sentences, there is no interpretable explanation for what each dimension of the vector space means. Therefore, unlike a data matrix where certain features correspond to protected attributes you want to take care of such as race or gender, a matrix of averaged word vectors has no such interpretation. However, by training our logistic regression algorithm on the data matrix leaving out certain principal components of the data and measuring the PLDB of the resulting model, we can figure out which parts of the data contain more or less toxicity bias. We name our method to find and use the principal components with the least amount of toxicity bias, Leave-One-Out-PCA or LOOPCA. We describe this method more formally below.

We perform ERM logistic regression on our $n \times m$ data matrix, with one singular vector removed at a time. The new ERM equation is

$$\min_{w \in R^d} \frac{1}{n} \sum_{i=1}^{n} l(y_i, w^T V_{\{-j\}} x_i) + \lambda \|w\|^2$$

where $V_{\{-j\}} x_i$ is x_i projected onto all but the j-th singular vector. After training our logistic regression algorithm for each leave one out component, we evaluate our model with PLDB and search for the components, that when left out, result in the most fair (lowest PLDB) algorithm. This allows us to not only to create fairer models with respect to PLDB but also allows us to audit principal components of our matrix, and add some meaning to the dimensions of the training data matrix. In our experiments, we show how leaving different principal components out change validation accuracy and PLDB. With respect to PCA, there are other methods we could employ, measuring PLDB for various permutations of principal components, but for this work we focus on Leave-One-Out-PCA.

4.3 Summary

In this section we present mitigation techniques for unintended demographic bias. For word embeddings, we show how we can use adversarial learning to re-embed textual protected attributes to be less correlated with sentiment or toxicity. For mitigation at the learning algorithm level, we showed how we can use regularization to bias a model to find solutions with a lower chance to unfairly discriminate. We also showed how we can use prediction level auditing, namely the PLDB metric, to search for principal components of the data that contain less toxicity bias. In our experiments we show how each of the methods perform in mitigating unintended demographic bias and creating fairer ML systems as a whole.

Chapter 5

Evaluation and Results

We evaluate our proposed techniques for understanding and mitigating unintended demographic bias. For word embeddings we evaluate the effectiveness of our RNSB framework in measuring sentiment bias. We also show that we can use adversarial learning to effectively mitigate unintended demographic bias in real NLP systems for sentiment regression and toxicity prediction. Finally, we show how we can mitigate unintended demographic bias at the learning algorithm level using CR and LOOPCA.

5.1 Word Embeddings

With respect to word embeddings, we discussed two important methods: measuring unintended bias with RNSB and mitigating unintended demographic bias with respect to toxicity and sentiment using adversarial learning. Making ML systems fairer in practice requires a similar workflow: first, creating tools to better understand unintended bias, then crafting techniques to mitigate this bias to create fairer ML systems.

5.1.1 Measuring Word Embedding Sentiment Bias with RNSB

We now explore the uses of our RNSB framework. We look at two cases studies: National Origin Discrimination and Religious Discrimination. For each case study,

we create a set of the most frequent identity terms from the protected groups (national origin and religion in this case) in the Wikipedia word corpus. We run our framework on the set of identity terms for National Origin Discrimination and Religious Discrimination and analyze the results. First, we compare the RNSB metric on various word embedding datasets, showing that our metric is consistent with other word embedding analysis like WEAT [7]. We then show that our framework enables an insightful view into how our word embeddings are biased.

Models and Data

We evaluate three pretrained models: GloVe [36], Word2vec [28], and ConceptNet [41]. GloVe and Word2vec embeddings have been shown to contain unintended bias in works [5], [7]. ConceptNet has been shown to be less biased than these models [42]. This is due to the wide corpora of text used to train ConceptNet in addition to some techniques from [5] used to debias the model. As part of our RNSB framework, we also use the Sentiment Lexicon [19] to represent our gold standard unbiased training set. This dataset has been shown to be a trustworthy dataset for sentiment analysis [34, 26, 44]. We trust these labels to be unbiased so that we may isolate the unintended biases entering our system to the word embeddings.

Interpretabillity in Word Embeddings through RNSB Framework

For this analysis, we vary the word embedding model used in our framework and calculate the RNSB metric for each embedding. Our results are displayed in Table 5.1. For both case studies, the unintended demographic bias is highest in GloVe, as shown by the largest RNSB metric. Word2Vec, having a smaller RNSB, still contains a fair amount of unintended bias. Although the RNSB metric is not directly comparable to WEAT scores, these results are still consistent with some of the unfairness predicted by [7]. The WEAT score shows that word embeddings like Word2vec and GloVe are biased with respect to national origin because European-American names are more correlated with positive sentiment than African-American names. Our metric captures the same types of biases, but has a clear and larger scope, measuring

discrimination with respect to all identifies in a protected group. Furthermore our metric is able to capture that ConceptNet has much less unintended social bias than GloVe or Word2vec. ConceptNet [41] is a state of the art model that mixes models like GloVe and Word2vec, creating fairer word embeddings. Through the RNSB metric, you can see that the bias and unfairness of these word embeddings are an order of magnitude lower than than GloVe or Word2vec.

Case Study	**GloVe**	**Word2Vec**	**ConceptNet**
National Origin Identity	0.6225	0.1945	**0.0102**
Religion Identity	0.3692	0.1026	**0.0291**

Table 5.1: Table showing our RNSB metric for various word embeddings on two case studies. Our metric effectively predicts the bias in the presented word embeddings with respect to negative sentiment.

Using the probability distribution of negative sentiment for all the identity terms in a protected group, we can compare how fairer word embeddings like ConceptNet equalize negative sentiment. Figure 5-1 shows three histograms dealing with negative sentiment. As described earlier, embeddings without unintended demographic bias is achieved when each identity term within a protected group has equal negative sentiment. The top two histograms show the negative sentiment probability for each identity normalized across all terms to be a probability distribution. The top histogram is computed using the GloVe word embeddings, and the second histogram is computed using the fairer ConceptNet embeddings. In the third histogram, we compute the delta between the GloVe and ConceptNet histograms and can see how ConceptNet is adding more relative positive sentiment (green) or removing it (red) to achieve a more uniform distribution. This type of analysis is very insightful as it enables one to see which identities are more at risk for discrimination than others, and how positive sentiment can be added to equalize the relative negative sentiment. For example, in comparison to GloVe, ConceptNet seems to add a lot of relative positive sentiment to the *Indian* and *Mexican* identity terms.

A more direct way to measure how certain groups receive similar unfair treatment is to compute a correlation matrix between the vectors containing negative sentiment predictions for each identity term. We compute this matrix for the same two cases:

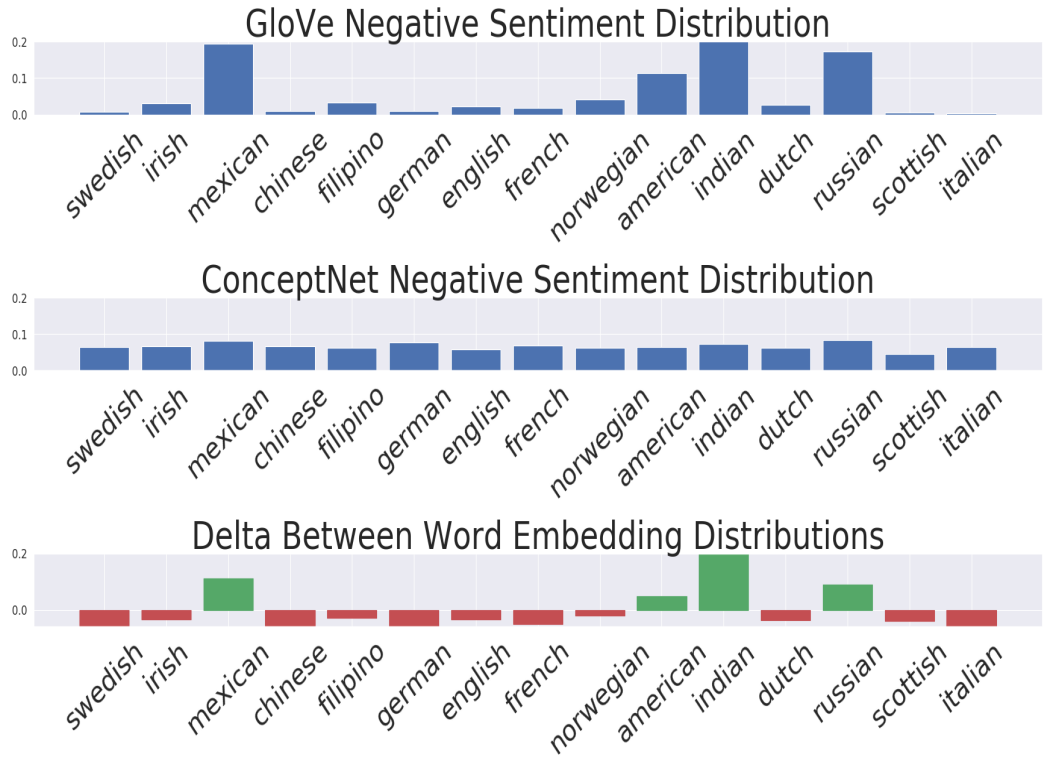


Figure 5-1: Histograms showing relative negative sentiment probability between national origin identity terms. The top graph is GloVe, then ConceptNet underneath. We present the delta between these two histograms showing that ConceptNet adds more positive sentiment (green) to help equalize the negative sentient between identities.

GloVe word embeddings (left) and ConceptNet word embeddings (right) shown in Figure 5-2. The GloVe word embedding correlation matrix contains a lot of dark low correlations between identities, as a lot of identities contain small amounts of negative sentiment. But this visual brings out that certain groups like *Indian*, *Mexican*, and *Russian* have a high correlation, indicating that they could be treated similarly unfairly. For the ConceptNet word embeddings, we see a much more colorful heat map, indicating there are higher correlations between more identity terms. This hints that ConceptNet contains less targeted discrimination via negative sentiment. This visual also brings out slight differences in negative sentiment prediction. Identity terms like *Scottish* have lower correlations across the board, manifesting that this identity has slightly less negative sentiment than the rest of the identities. This is important to analyze to get a broader context for how various identities could receive different

amounts of discrimination stemming from the word embedding bias.

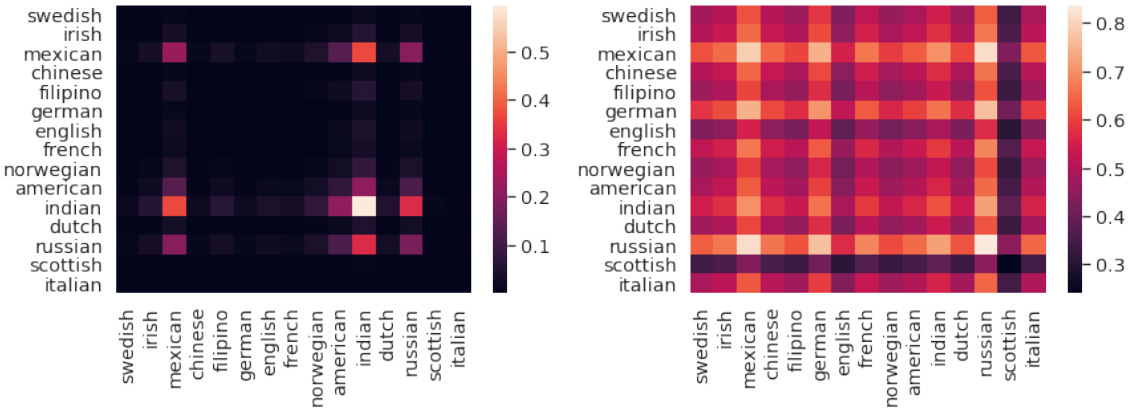


Figure 5-2: National origin correlation matrix for negative sentiment prediction using GloVe (left) and ConceptNet (right) word embeddings. We can use these figures to analyze how certain groups could be similarly discriminated against via their negative sentiment correlation.

Because our RNSB metric has a realistic sentiment classifier baked in the computation, we can also easily observe how fairness is impacted by modifying the learning function itself. This is another potential use of our framework as it allows quick iteration for testing and evaluating ways to mitigate word embedding bias in the learning algorithm itself. As an example, we evaluate how kernel methods can be used to mitigate the bias in word embeddings with respect to the sentiment prediction task, which is used for common tasks like a movie sentiment analyzer [33]. For the National Origin Identity case study and GloVe word embeddings, we now use a probabilistic support vector classifier (SVC) and vary the kernel, reporting our findings for the RNSB metric in Table 5.2.

	Original	**Sigmoid**	**Polynomial**	**RBF**
RNSB	0.6225	0.4665	0.5677	0.4009
Test Accuracy	0.9071	0.9087	0.7067	0.9135

Table 5.2: Table showing our RNSB metric for various kernels in the sentiment classifier. We can limit unintended bias coming from the word embeddings for the sentiment prediction task by using various kernels, without impacting test accuracy.

One can see how some of the unintended bias from the word embeddings is mitigated through the use of different kernels. Kernels are essentially reshaping the input

vector space on the fly, so it is very likely that they distort some of the unwanted correlations between identity terms and negative sentiment terms. As seen in the table, one can use an RBF kernel in an SVC and not only improve accuracy but also lower unfairness with respect to our RNSB metric. Some kernel methods like the polynomial kernel sacrifice too much performance for more fairness. Through our framework you can easily inspect the degree to which one can mitigate the unfairness coming from the word embeddings. In the next section we see our results for more directly mitigating unintended demographic bias at the word embedding level.

5.1.2 Mitigating Word Embedding Bias Leads to Fairer ML Algorithms

In the last section, we quantified the degree to which word embeddings contain unintended demographic bias. With this in mind we can now try to mitigate some of this bias. In this section we employ the use of our adversarial algorithm to debias demographic word vectors with respect to sentiment and toxicity for realistic NLP systems. We show our adversarial debiasing technique for word embeddings is robust and effective at making downstream machine learning decision fairer. We first discuss our results with respect to applications ins sentiment analysis, then discuss results with respect to toxicity prediction.

Mitigating Sentiment Bias

We evaluate the effectiveness of our method in minimizing the sentiment polarity of a word vector without distorting the word vector's semantic meaning within the vector space. We first describe the datasets used in our experiments. Next, we evaluate the improved fairness in the depolarized word embeddings for sets of identity terms. Finally, we take a look at how our debiased word embeddings make a downstream machine learning task fairer.

Template	#sent.
Sentences with emotion words:	
1. <Person> feels <emotional state word>.	1,200
2. The situation makes <person> feel <emotional state word>.	1,200
3. I made <person> feel <emotional state word>.	1,200
4. <Person> made me feel <emotional state word>.	1,200
5. <Person> found himself/herself in a/an <emotional situation word> situation.	1,200
6. <Person> told us all about the recent <emotional situation word> events.	1,200
7. The conversation with <person> was <emotional situation word>.	1,200

African American		European American	
Female	**Male**	**Female**	**Male**
Ebony	Alonzo	Amanda	Adam
Jasmine	Alphonse	Betsy	Alan
Lakisha	Darnell	Courtney	Andrew
Latisha	Jamel	Ellen	Frank
Latoya	Jerome	Heather	Harry
Nichelle	Lamar	Katie	Jack
Shaniqua	Leroy	Kristin	Josh
Shereen	Malik	Melanie	Justin
Tanisha	Terrence	Nancy	Roger
Tia	Torrance	Stephanie	Ryan

Figure 5-3: *Left*: Template sentences from the EEC. *Right*: Demographic identity terms used to represent sensitive attributes in the sentences. We use this dataset to evaluate the fairness of realistic sentiment analysis systems.

Datasets for Experiments For our experiments, we need a set of positive and negative sentiment words to create our sentiment subspaces. We use the Sentiment Lexicon dataset from [19] to supply the necessary words. The word embedding models that we debias are the word2vec [28] pre-trained model trained on a large corpus of Google News data and GloVe word embedding [36] trained on a large Wikipedia corpus. Recent studies have shown that these pre-trained models contain many types of bias [5, 7]. The focus of this work is on minimizing the sentiment bias contained in the embeddings with respect to demographic identity terms. To evaluate the embedding model performance before and after debiasing, we use WordSim353 similarity dataset developed by [1]. It is difficult to comprehensively evaluate a word embedding model, but correlations between word vector similarities and human-assigned similarity judgments from WordSim353 gives us an insight into usefulness of our debiased word embeddings. Finally for our downstream application, we investigate models trained on the valence regression dataset from *SemEval-2018 Task 1 Affect in Tweets* [29]. We evaluate fairness in our models via benchmarks developed with the Equity Evaluation Corpus (EEC) from [24]. The EEC contains many template sentences that vary in the demographic identity terms used. Figure 5-3 shows examples of such template sentences along with names that tend to belong to African American or European American demographic groups. These names serve as the protected attributes between which we wish to measure discrimination.

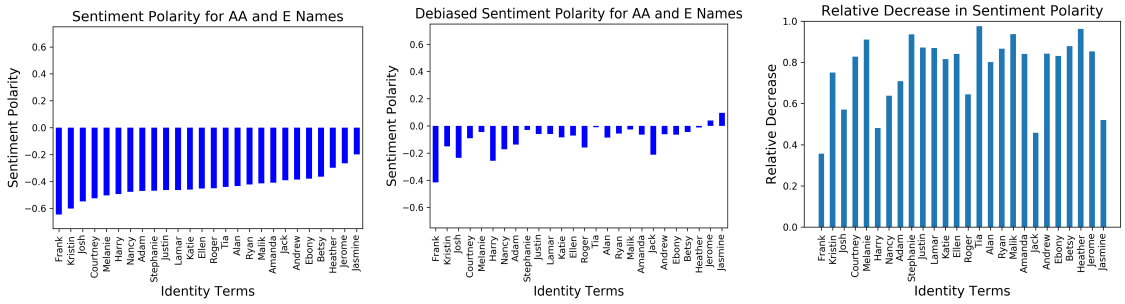


Figure 5-4: Histograms showing sentiment polarity for names that tend to belong to African American (AA) and European (E) demographics. The histograms to the far right shows the relative decrease in sentiment polarity after debiasing the identity term word vectors. After debiasing, we can reduce the overall sentiment polarity for this type of demographic attribute.

Fairness in Depolarized Word Embeddings We trained our algorithm on the first two million words from the word2vec embeddings and four hundred thousand words from GloVe to find the best way to reduce sentiment bias for each model. The use case of our algorithm is to only use the trained model to re-embed words that should not have sentiment polarity like demographic attributes. Therefore, we evaluate the sentiment polarity of demographic identity terms before and after debiasing. Demographic attributes can be represented in text in many ways (i.e gendered words like *he, she*, religious identifiers like *Catholic, Jewish*, or even names that tend to belong to certain race). We can effectively reduce sentiment bias for many demographic attributes in text. A quick way to measure the total amount of sentiment bias is to sum the absolute value of the projections onto the sentiment vector for the demographic identity terms. We call this sum, the *summed sentiment polarity*. Table 5.3 shows the relative decrease of *summed sentiment polarity* for three different types of demographic attributes: gendered terms, names, and religious identifiers. The names and gendered terms are from [24] where they are used to study sentiment bias and the religion identity terms we use are the top 15 most popular religious identifiers in the world. It is important to note that in a real world use case, a practitioner would pick the exact terms to debias using our algorithm.

The lower the *summed sentiment polarity*, the more effectively we can remove sentiment bias for that type of demographic attribute. We examine how this lower

sentiment bias actually makes downstream sentiment analysis algorithms fairer for names and gendered words in later experiments. Figure 5-4 displays histograms describing the projection of identity term word vectors describing typical names from European and African American demographics onto the directional sentiment vector, before and after debiasing. The resulting number is the sentiment polarity for a given word vector. We also show the relative decrease in sentiment polarity in the far right plot. Although the rates of sentiment bias decrease are not equal for every term, we still attain an average relative decrease across the names of 76.8%. This makes it much less likely for a downstream sentiment analysis algorithm to pick up on the sentiment bias. It is also worth while to note that most demographic identity terms tend to naturally be polarized towards the negative subspace. This is however, irrelevant as our goal is to re-embed word vectors without positive or negative sentiment polarity. Furthermore, we want to verify our models' ability to generally reduce sentiment bias for new words, not seen in the training set. To this end, we removed the identity terms used in this experiment from the training set. We trained our models for 40,000 steps with batch size of 1000 words. The models were trained with an adversarial weight, $\alpha = .5$.

	Gender	**Names**	**Religions**
Relative Decrease in Sentiment Bias: GloVe	79%	53%	45%
Relative Decrease in Sentiment Bias: Word2Vec	59%	76%	63%

Table 5.3: Table showing relative decrease in summed sentiment polarity for identity terms for three types of demographic attributes: *Gender*, *Names*, and *Religions* after applying our algorithm. For popular pre-trained models GloVe and Word2Vec, we can effectively reduce sentiment bias within the embedding vector space.

Post-Debiasing Word Embedding Performance As our model moves word vectors around in the embedding's vector space, we also run the risk of distorting the vectors, possibly losing their semantic relation to the words around them. To evaluate the debiased word embeddings with respect to their relation to other words, we can use notions like analogy completion tasks or similarity measures to surrounding word vectors. After debiasing word vectors like *man*, we still retain the word vector analogy, man:woman as boy:girl. We analyze our debiased embeddings more formally on the

word similarity dataset, WordSim353 [1]. The dataset is composed of pairs of words with labeled human similarity judgments. Using our trained model, we debias one word from each pair, evaluate the resulting cosine similarity and compute spearman correlation with the human judgments in WordSim353. The spearman correlation serves to measure the effect debiasing one word has on its semantic relation to another word from the embeddings. We evaluate spearman correlation score for 6 different models trained with differing adversarial weights, α, and compare to our the *Summed Sentiment Polarity* for the set of names presented in Figure 5-4. Figure 5-5 shows the spearman correlation vs sentiment bias for 6 settings of α. The spearman correlation is barely changed by debiasing the word vectors. This is likely due to the fact that our loss function L_p constrains a debiased word vector from straying too far away from its original place in the vector space.

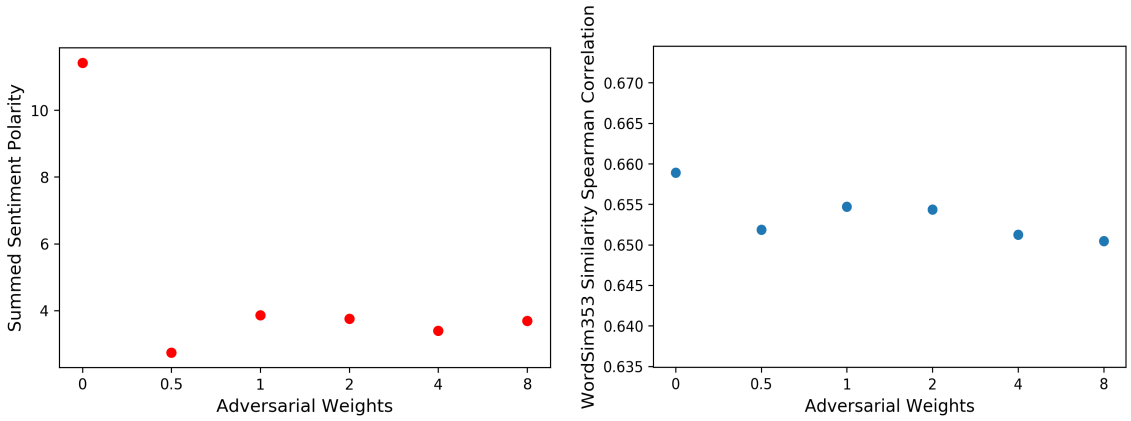


Figure 5-5: *Left*: Plot summarizing how sentiment polarity for the set of names presented in Figure 5-4 varies for different adversarial weights. *Right*: Plot showing how the WordSim353 word embedding performance varies with different settings of our adversarial weight.

Downstream Sentiment Valence Regression We evaluate how our debiased word vectors make downstream tasks in sentiment analysis less discriminatory. There are many different ways to frame and measure sentiment in a sentence (i.e positive/negative, anger, sadness, valence). We focus on the task of regressing sentiment intensity or valence. [24] investigates the unfairness in this type of sentiment analysis task for over 200 different models trained on the SemEval-2018 Task 1 Affect

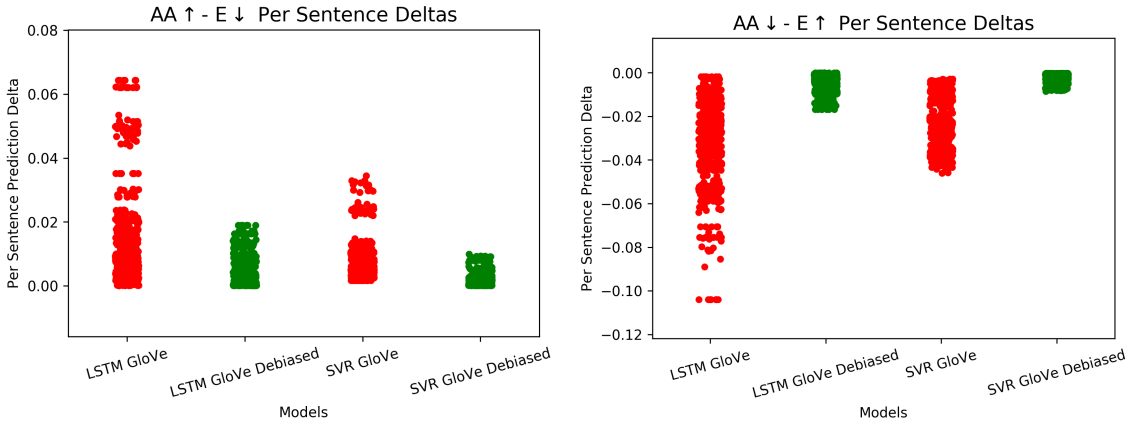
in Tweets [29]. The authors create the EEC dataset to help measure differences in valence predictions between similar sentences that differ in the presence of a demographic identity term. The authors measure unfairness in valence regression for race using names that tend to belong to the African American demographics vs European demographics. For example, a sentence template from the EEC dataset looks like *<Name> feels <emotional state word>*. For gender, the authors measure unfairness in valence regression using gendered words like he or she. Similar templates are used in this scenario (*<He/she> feels <emotional state word>*.) The authors perform valence predictions on sentences in the EEC database with emotional state words, and compare the average scores between different demographic groups. We describe the comparison metrics below.

- Avg. score difference **AA↑-↓E**: "The average Δ for only those pairs where the score for the African American noun phrase sentence is higher. The greater the magnitude of this score, the stronger the bias in systems that consistently give higher scores to African American-associated sentences."

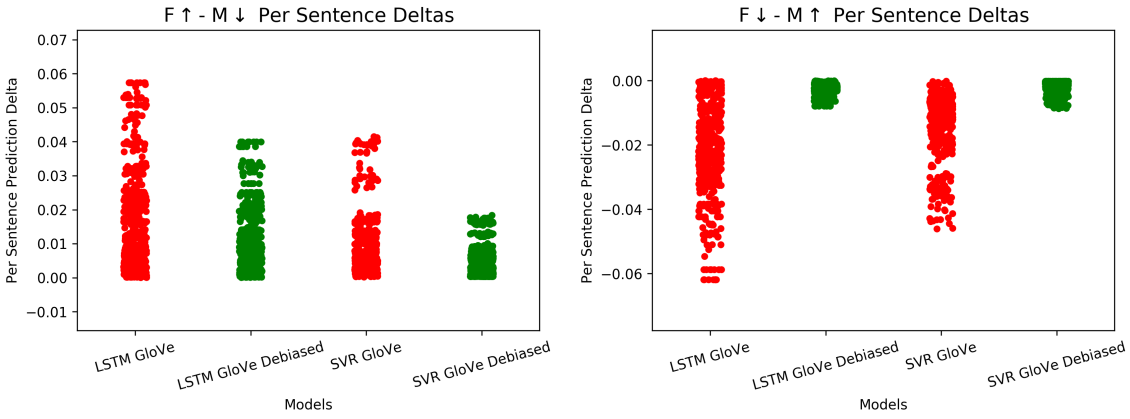
- Avg. score difference **AA↓-↑E**: "The average Δ for only those pairs where the score for the African American noun phrase sentence is lower. The greater the magnitude of this score, the stronger the bias in systems that consistently give lower scores to African American-associated sentences."

The same metrics are used to compare Female (F) and Male (M) sentences (**F↑-↓M** and **F↓-↑M**). We train a classical and deep regression model on the valence regression training set from SemEval-2018 Task 1 Affect in Tweets. As noted in [24], different model choices can result in varying degrees of bias. For example, a deeper model might be more sensitive to the subtle biases that enter either through the word embedding models or training set. Details for the two models are listed below.

- The classical model we use is a support vector regression (SVR) model. We encode text using the GloVe word embeddings trained on the Wikipedia corpus [36] and average the word vectors in the sentence. We feed the resulting vectors into the SVR algorithm to predict a valance score between 0 and 1.



(a)



(b)

Figure 5-6: Scatter plots showing the distribution of sentiment valence regression score differences for 4 demographic group comparisons. For each group we measure score deltas for 4 different models (two classical, two deep) using our debiased word embeddings and original word embedding model. In every category, our debiased word embedding models (shown in green) minimizes the bias between demographic groups with respect to sentiment valence predictions for similar sentences.

- The deep model we use is an recurrent model with 128 LSTM units followed by a dense layer with 64 units. We use another dense layer with one unit to output a valence prediction between 0 and 1. We use the mean squared error loss function. Finally, we encode text using the same GloVe word embeddings and represent a sentence as a padded matrix of word vectors which we pass into our model.

	Avg.Δ: AA↑-↓E	**Avg.Δ: AA↓-↑E**	**Avg.Δ:F↑-↓M**	**Avg.Δ:F↓-↑E**
LSTM GloVe	0.0159	0.0335	0.0166	0.0220
LSTM GloVe Debiased	0.0042	0.0046	0.0120	0.0024
Relative Bias Decrease	**73%**	**87%**	**28%**	**80%**
SVR GloVe	0.0085	0.0203	0.0098	0.0140
SVR GloVe Debiased	0.0021	0.0024	0.0054	0.0019
Relative Bias Decrease	**75%**	**90%**	**45%**	**86%**

Table 5.4: Table showing sentiment bias measures for 4 groups, on 4 different models. When training on our debiased word embeddings, we can achieve up to a 90 % decrease in unintended demographic bias via listed metrics.

For each model, we train with the original GloVe word embeddings and debiased GloVe word embeddings using our algorithm. The resulting pearson correlations scores on the SemEval-2018 Task 1 gold standard set was 42%, 43% for the GloVe and Debiased GloVe SVR models respectively and 59% and 61% for the GloVe and Debiased GloVe LSTM models respectively. Using our word embeddings with debiased demographic attributes actually improved the regression performance. This is further proof that we do not have adverse side effects in performance when attempting to make the word embedding models fairer. We also compare fairness for the four models via the avg. score differences for the groups: **AA↑-↓E**, **AA↓-↑E**, **F↑-↓M**, **F↓-↑M**, shown in Table 5.4. We see that for every category, our debiased word embeddings result in a smaller average gap between similar sentences with different demographic identity terms. To get a better sense for the distribution of pairwise valence score deltas for sentences with AA names vs E names and Male vs Female identity terms, we plotted a scatter plot in Figure 5-6. The dots are sentiment valence differences between the demographic groups for a particular sentence. Red dots are using the original word embedding models and green dots are using our debiased word embeddings. For every category, our algorithm compresses the variance in valence

score deltas, showing that we have effectively reduced sentiment bias in a real world task. The remaining bias seen after using our word embeddings could come from other sources like unbalances in the tweets dataset or choice of learning algorithm. But we have shown that most of the sentiment bias for this task can be removed with our debiased word embeddings.

Mitigating Toxicity Bias

We evaluate the effectiveness of our method in minimizing the toxicity polarity of a word vector without distorting the word vector's semantic meaning within the vector space. We first describe the datasets used in our experiments. Next, we evaluate the improved fairness in the depolarized word embeddings for a set of identity terms, and examine the trade off between fairness and our word embedding's semantic meaning. Finally, we show that our word embeddings make the toxicity classification algorithm used in [10] much fairer.

Models and Data For our experiments, we need a set of positive and negative sentiment words to create our toxicity subspaces. We use the Sentiment Lexicon dataset from [19] to supply the necessary words. We investigate debiasing demographic identity terms from both the pre-trained Word2vec model trained on a large corpus of news and the GloVe word embeddings trained on a large Wikipedia corpora. The focus of this work is on minimizing the toxicity polarization contained in the word embeddings with respect to demographic identity terms. Additionally, to evaluate the usefulness of the word embeddings before and after debiasing, we use WordSim353 similarity dataset developed by [1]. Finally, we compare how our debiased word embeddings perform out in the wild on 127,820 Wikipedia Talk Page comments that were labeled by human raters as non-toxic or toxic. We train the same Convolutional Neural Network models presented in [10], with and without our debiased embeddings and compare the resulting fairness. We evaluate fairness using the Pinned AUC metric and synthetic dataset developed in [10]. Figure 5-7 shows sentence template samples from this synthetic dataset along with a list of demographic identity terms

Template Examples	Label
I am <IDENTITY>	Non-Toxic
I am a <IDENTITY> person, ask me anything	Non-Toxic
<IDENTITY> people are just like everyone else	Non-Toxic
I hate all <IDENTITY>	Toxic
I am a <IDENTITY> person and I hate your guts and think you suck	Toxic
<IDENTITY> people are gross and universally terrible	Toxic

Term	Toxic	Overall
atheist	0.09%	0.10%
queer	0.30%	0.06%
gay	3%	0.50%
transgender	0.04%	0.02%
lesbian	0.10%	0.04%
homosexual	0.80%	0.20%
feminist	0.05%	0.05%
black	0.70%	0.60%
white	0.90%	0.70%
heterosexual	0.02%	0.03%
islam	0.10%	0.08%
muslim	0.20%	0.10%
bisexual	0.01%	0.03%

Figure 5-7: *Left*: Sentence templates from the synthetic evaluation set developed in [10]. *Right*: Percentage of the time identity terms appear in toxic sentences vs toxic and non toxic sentences together.

used to represent protected attributes within the sentence. The templates are very similar to the EEC. The right image in Figure 5-7 shows the percentage of the time a certain identity term show up in toxic vs all comments. This is a measure of bias within the Wikipedia Talk Page dataset. [10] corrects for the differences in toxic sentence representation for certain identity terms by adding synthetic sentences to balance out the dataset. We compare how this type of bias mitigation at the dataset level compares to bias mitigation at the word embedding level in terms of the overall fairness of the toxicity classification algorithm.

Fairness in Depolarized Word Embeddings We trained our adversarial learning algorithm on the first two million words from the Word2vec or four hundred thousand words from the GloVe embeddings to find the best way to reduce a word vector's correlation with toxicity. The use case of our algorithm is to only use the trained model to re-embed words that should not have toxicity polarity like demographic identity terms. Therefore, we evaluate the toxicity polarity of identity terms before and after debiasing. Furthermore, we want to verify our model's ability to generally reduce toxicity polarity for new words, not seen in the training set. To this end, we removed the identity terms used in this experiment from the training set. We trained our model for 30,000 steps with batch size of 1000 words. Similar to the sentiment case, we analyze the toxicity polarity of demographic identity word vectors

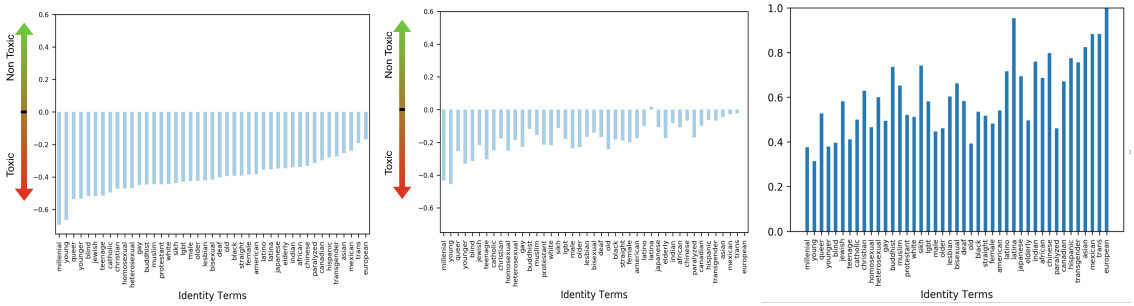


Figure 5-8: *Left*: Word2vec word vector toxicity polarity for a set of demographic identity terms. *Middle*: Toxicity polarity after applying our technique to the identity term word vectors. *Right*: Relative decrease in magnitude of toxicity polarity, with an average decrease of 60%.

before and after depolarizing with our trained model. Our results for the 50 demographic identity terms presented in [10] is presented in Figure 5-8. The projection onto the directional toxicity vector is the toxicity polarity for a given word vector. We see that the histogram bars are significantly closer to 0 (no toxicity projection) after debiasing. Though different terms have different rates of depolarization after debiasing, our approach leads to an average term-wise decrease of toxicity polarization of 60%. The model used for these figures was trained with an adversarial weight, $\alpha = .5$.

	Summed Toxicity Polarity
GloVe Word Embeddings	16.62
Debiased GloVe Word Embeddings	6.60
Word2Vec Word Embeddings	17.45
Debiased Word2Vec Word Embeddings Treatment	7.47

Table 5.5: Table showing summed toxicity polarity for 50 demographic identity terms for two word embedding models before and after debiasing via our technique. We can debias the GloVe and Word2vec word embeddings by 60.2% and 57.2% respectively.

Similar to summed sentiment polarity, we also look at summed toxicity polarity of the 50 demographic identity term set before and after depolarization. Summed toxicity polarity is the sum of toxicity polarity magnitudes for the terms within the set. With respect to summed toxicity polarity, we can debias the GloVe and Word2vec word embeddings by 60.2% and 57.2% respectively. Table 5.5 displays the summed toxicity polarity metric for GloVe and Word2vec embeddings before and after debiasing identity terms. Interestingly, the numbers are pretty similar, suggesting that our

algorithm is practically useful and stable for various NLP applications using different pretrain embeddings. Next, we evaluate how we can not only drastically decrease toxicity polarity, but do so in a way that preserves the meaning of the word vectors.

Trading off Word Embedding Performance with Toxicity Polarization As our model moves word vectors around in the embedding vector space, we also run the risk of distorting the vectors, possibly losing their semantic relation to the words around them. To evaluate the debiased word embeddings with respect to their relation to other words, we can use notions like analogy completion tasks or similarity measures to surrounding word vectors. After debiasing word vectors like *man*, we still retain the word vector analogy, man:woman as boy:girl. Additionally, debiased word vectors retain the same relationships to the words around them. Figure 5-9 shows the sorted output of the 10 nearest neighbors, via cosine distance, for the demographic identity term, *male*, before and after debiasing with our technique. Though some similarity measures differ and various words are reordered, the debiased word vectors generally retain the same relationship to the surrounding word vectors.

We also analyze our debiased embeddings more formally on the word similarity dataset, WordSim353 [1]. The dataset is composed of pairs of words with labeled human similarity judgments. Using our trained model, we debias one word from each WordSim353 pair, evaluate the resulting cosine similarity and compute spearman correlation with the human judgments in WordSim353. The spearman correlation serves to measure the effect debiasing one word has on its semantic relation to other unchanged words in the embeddings.

We evaluate the WordSim353 spearman correlation score for 6 different models trained with differing adversarial weights, α, and compare to our measure of summed toxicity polarity for the demographic identity terms in our study. The left plot in Figure 5-10 shows the WordSim353 spearman correlation for 6 settings of α. The right plot in Figure 5-10 shows the summed toxicity polarity for the 6 different settings of α. By weighting the adversarial objective, we can effectively minimize toxicity polarity, without distorting the word embeddings via the WordSim353 similarity correlation.

K=10 Nearest Neighbors Before Debiasing

```
[('male', 1.0000001192092896),
 ('female', 0.8405333161354065),
 ('males', 0.7579617500305176),
 ('females', 0.7030534744262695),
 ('accomplice_Hudgens', 0.6375256776809692),
 ('Male', 0.6288970112800598),
 ('Female', 0.5971192121505737),
 ('femal', 0.5850157737731934),
 ('Caucasian_males', 0.5570127367973328),
 ('masculinised', 0.5340930223464966)]
```

K=10 Nearest Neighbors After Debiasing

```
[('male', 0.9783856272697449),
 ('female', 0.8173885941505432),
 ('males', 0.7334728837013245),
 ('females', 0.6692583560943604),
 ('Male', 0.60198974609375),
 ('Female', 0.5653233528137207),
 ('femal', 0.5433569550514221),
 ('Caucasian_males', 0.5221525430679321),
 ('accomplice_Hudgens', 0.5165694952011108),
 ('heterosexual_males', 0.4910895824432373)]
```

Figure 5-9: Top 10 nearest neighbors for the demographic identity term, *male* (bolded), before and after debiasing using our technique. There is not much distortion of the word vectors relationship to its neighbors after decorrelating with toxicity.

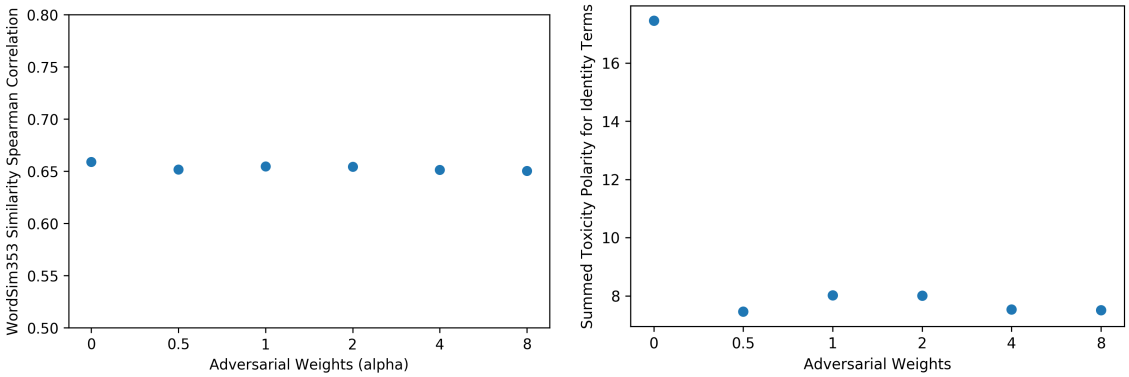


Figure 5-10: *Left*: Plot showing how the WordSim353 word embedding performance varies with different settings of our adversarial weight. *Right*: Plot summarizing how toxicity polarity for a set of demographic identity terms is decreased for various adversarial weights. We can drastically decrease the summed toxicity polarity for a set of identity terms with our adversarial objective.

This is likely due to the fact that we have a loss function L_p that constrains the predicted word vectors from straying too far away from its original place in the vector space.

Thus far in this analysis, we have discussed our method for reducing toxicity polarization or correlation for demographic identity terms, but have left out explicitly talking about fairness. In the next section we evaluate how our debiased embeddings can be used in a practical NLP task to create fairer classifiers.

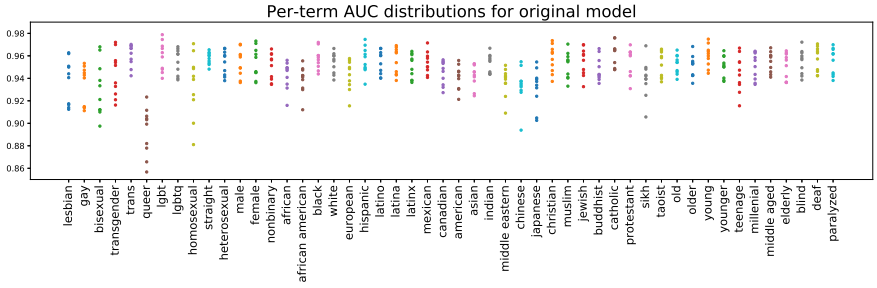
Debiased Word Vectors Make Fairer Downstream Toxicity Classifier We evaluate how our debiased word vectors make a downstream task less discriminatory.

As mentioned earlier in this work, toxicity classification is used in many scenarios such as censoring abusive or offensive comments on online forums. Due to possible unintended biases surrounding textual protected attributes like demographic identity terms, toxicity classification algorithms risk discriminatory censoring of various demographics. [10] has made great strides on mitigating toxicity bias at the dataset level, training a classification algorithm on a preprocessed dataset of Wikipedia Talk Page comments. The dataset is preprocessed to mitigate dangerous over representations of certain demographic identity terms in toxic sentences. This helps the trained model make fewer false positive predictions for nontoxic sentences that contain a demographic identity term. To formally measure fairness for this application, the authors propose a metric called *Pinned AUC Equality Difference*, which measures area under ROC curve for a balanced dataset of toxic and nontoxic synthetic sentences containing a particular identity term, t, from a set, T. The *Pinned AUC Equality Difference*, $pAUCed$, metric is presented below for convenience.

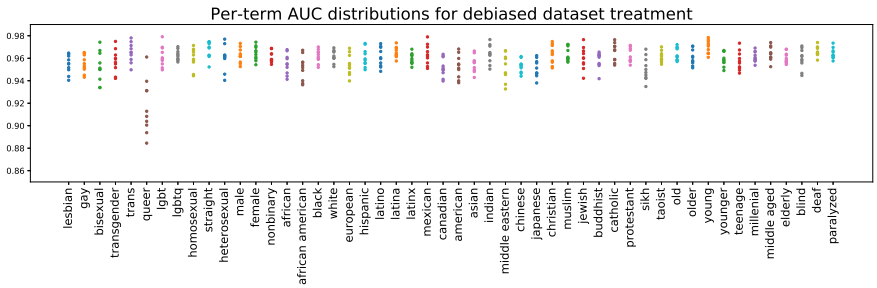
$$pAUCed = \sum_{t \in T} |AUC - pAUC_t|$$

Where AUC is the model's overall AUC and $pAUC_t$ is the AUC for the sentences containing the particular identity term, t.

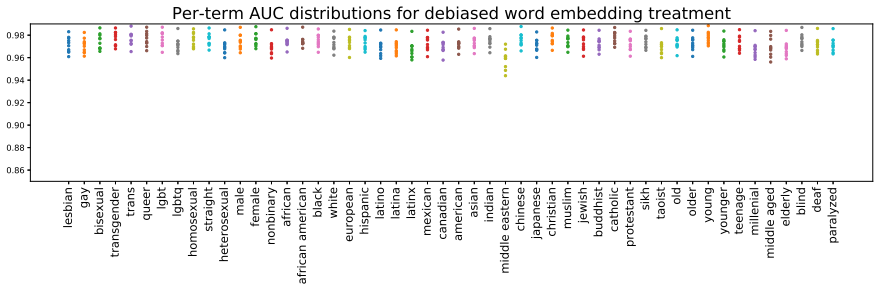
Although rebalancing the training set helps the model make fewer dangerous false positive predictions, it misses much of the bias that stems from unfair toxicity correlations in the word embeddings. Figure 5-11 shows AUC results for 10 training runs of a Convolution Neural Network evaluated on the synthetic template dataset for each identity term. The top figure (a) is the CNN trained on the unbalanced Wikipedia Talk dataset with no debiasing treatment. (b) shows the results for the CNN trained on the balanced dataset via the method in [10]. Between these two graphs, one can see that there is less variance between the models performance on different identity terms, indicating less discriminatory behavior. We get even less variance for (c), showing the results for the model trained on the unbalanced Wikipedia Talk dataset with our detoxified identity term word vectors. Furthermore, we get even better re-



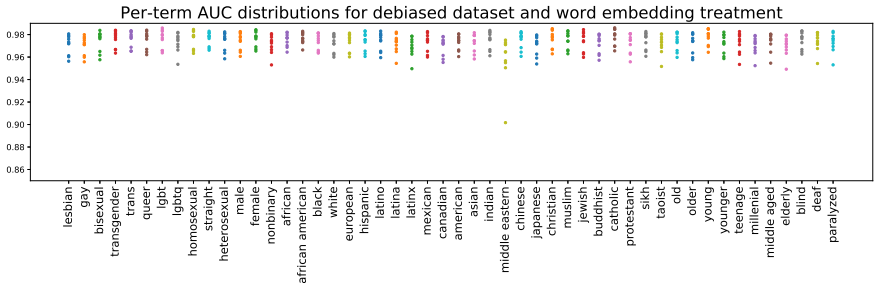
(a)



(b)



(c)



(d)

Figure 5-11: *a*: Results for the toxicity classifier with no debiased treatment. *b*: Results for the model trained on the debiased dataset technique from [10]. *c*: Results for the model trained with our debiased word embeddings. *d*: Results for the model trained with both the debiased dataset and debiased word embedding treatment. Graphs were generated using open source code: https://github.com/conversationai/unintended-ml-bias-analysis. Our technique, (c) does a better job of minimizing AUC discrepancies between various demographic groups than the dataset debiasing technique (b).

sults when combining the dataset debiasing technique with our word embeddings (d). Confirming these qualitative results more formally, Table 5.6 shows the *Pinned AUC Equality Difference* metric for the 4 different types of model treatments. When just using our debiased word embeddings, we get a 52% increase in fairness via the *Pinned AUC Equality Difference* metric over the dataset debiasing technique developed in [10]. However, when combining this technique with our debiased word embeddings, we get the best results with a 59% percent increase in fairness via the *Pinned AUC Equality Difference* metric over the dataset debiasing treatment. It is important to realize, that though our word embedding debiasing does a better job at creating a fairer model, it is mitigating a different source of bias. It is very possible that word embeddings bias have a larger impact on model fairness that dataset bias. Still, we saw improved fairness when applying mitigation treatments at multiple levels of the NLP pipeline. Because unintended bias can enter a NLP pipeline at many different points, to achieve the fairest decision systems, we need to mitigate bias at multiple levels.

	Pinned AUC Equality Difference
Original Model	5.900
Debiased Dataset Treatment	3.756
Debiased Word Embedding Treatment	1.768
Debiased Dataset and Word Embedding Treatment	1.534

Table 5.6: *Pinned AUC Equality Difference* metric for 4 different model debiasing treatments for toxicity classification. Using our debiased word embedding model we get a 70% increase in fairness compared to no debiased treatment, and a 52% improvement over the debiased dataset treatment baseline.

5.2 Learning Algorithm

In Section 4.2 we explored two methods for mitigating toxicity bias at the learning algorithm level, Concept Regularization (CR) and Leave-One-Out-PCA (LOOPCA). As previously mentioned, mitigation techniques at this level have to deal with bias not only from the learning algorithm, but also from all the preceding steps (i.e. word embeddings, dataset). Unfortunately, this comes with a lack of specificity into what

types and sources of biases are being mitigated. Still, with metrics like our Prediction Level Demographic Bias (PLDB), we can get a high level estimate for the amount of demographic bias accumulated in the machine learning system.

5.2.1 Mitigating Toxicity Bias at the Learning Algorithm Level

We investigate how to reduce toxicity bias with respect to identity terms from a protected group. We focus on identity terms from three separate protected groups: *Race*, *Religion*, and *Gender*. For our experiments, we train our logistic regression algorithm on the Wikipedia Talk pages toxicity dataset [45]. We first show how we can effectively regularize the logistic regression algorithm, using CR, to contain less toxicity bias with moderate decreases in model performance. We then show how we can use PLDB to search for the principal components of our dataset with the most toxicity bias and leave them out in training our algorithm. Finally, we make comparisons between the various methods showing the strengths of mitigating toxicity bias using CR and LOOPCA.

Models and Data

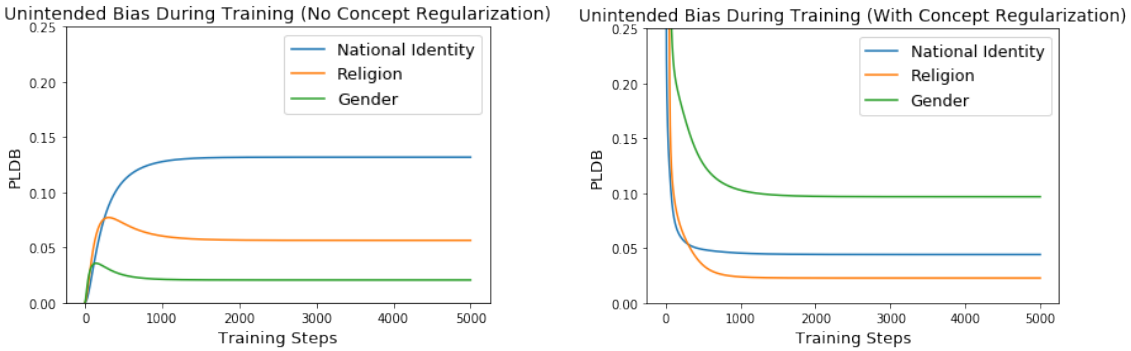
For this work, one of the sources of toxicity bias we are trying to mitigate is the GloVe [36] word embeddings model trained on the Wikipedia corpus. GloVe, along with other pre-trained word embeddings like Word2vec have been shown to contain unintended demographic bias in works [5, 7]. Another source of toxicity bias is our Wikipedia Talk pages toxicity dataset [45]. This dataset contains 127,820 Wikipedia Talk Page comments that were labeled by human raters as non-toxic or toxic. [10] has shown that this dataset contains unintended demographic bias, as certain demographic identity terms show up in a disproportionately large amount of toxic comments. Finally, we use a logistic regression algorithm to predict sentence toxicity. Though harder to pinpoint unintended bias introduced by the choice of learning model, we choose a simple model to lower risk of introducing more unin-

tended bias. We train our logistic regression model on a train-validation split from the Wikipedia Talk pages dataset. The dataset is vectorized by averaging the GloVe embeddings word vectors in each comment. To prevent unfair downstream discrimination, we aim to prevent demographic identity terms from contributing unequally to toxicity likelihood in our classifier.

Concept Regularization Reduces Toxicity Bias

As described earlier, PLDB serves as a way to measure differences in toxicity among identity terms, but doubles as a regularization term that we can directly minimize. It turns out that regularizing a model to decrease toxicity bias is very important as loss functions tend to be at odds with fairness measures like PLDB. Figure 5-12 manifests this notion. In the left image we see that PLDB naturally rises as our LR algorithm is trained without Concept Regularization. In the right image, we see that PLDB can be minimized with respect to each protected group using our CR. It is important to note that not all demographic group classes behave the same way under regularization. For example, CR actually resulted in a slightly higher PLDB for gender than the other two categories. This is likely due to the fact that gendered terms like *he/she* are used much more frequently in language than demographic terms like *American*. This can be corrected by raising the weight term β_j for the gender regularization term, at the expense of accuracy. This trade-off can be made by practitioner who could simply tune these corresponding β_j based on the constraints of the application.

Even though our regularization terms are not optimizing for performance, our final model accuracy is about 5% under the performance of the non-regularized model. This is a moderate decrease in accuracy that goes to show that fairness can often be at odds with accuracy. With the moderate decrease in accuracy, our model is capable of lowering its chances to discriminate with respect to multiple demographics at the same time. For our experiments, we set $\beta_j = .1$ for each regularization term and $\lambda = .01$. We also changed some of the identity terms between the terms included in our CR term and the PLDB measurements made on the model during training to show our method's ability to generalize to terms not used in the regularization.



(a) PLDB vs training steps without Concept Regularization

(b) PLDB vs training steps with Concept Regularization

Figure 5-12: Prediction Level Demographic Bias During Training

Leave-One-Out Principal Component Analysis

We used our PLDB metric to find the principal components of the data that are the most biased with respect to toxicity. For this experiment, we train our LR on our data matrix 20 times, each time leaving out a different principal component. Figure 5-13 displays a histogram of PLDB measures with respect to three protected groups for our 20 trained models. The gray line is the RNSB for the model with no components removed. Looking at Figure 5-13, we see that there is no component that contains all the bias. Toxicity bias is tied into multiple principal components of the data, further showing that toxicity bias is not easily removed. However, the second principal component tends to contain much of the toxicity bias. Unfortunately, as we show in Figure 5-14, this component contains a lot of the useful signal for determining toxicity. There are certain principal components of the data that, when left out, lead to slightly less biased results with small changes in model accuracy. A model practitioner would need to do a similar analysis to discover and omit principal components that have too much toxicity bias. Perhaps unsurprisingly, the effect on PLDB seems to decrease the smaller the singular value (principal component greater than 11), as it is known that smaller singular values/vectors contain less signal and more noise. We explore the relationships between PLDB and accuracy in the next section.

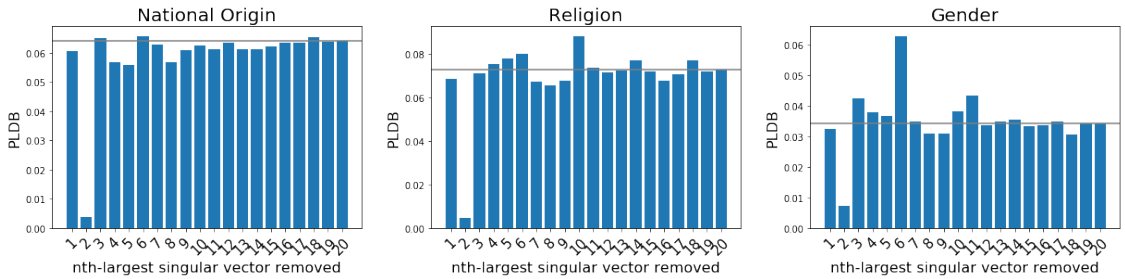


Figure 5-13: PLDB for each singular vector in leave-one-out-PCA (LOOPCA). The gray line indicates the non-regularized PLDB for logistic regression.

5.2.2 Accuracy vs Toxicity Bias

We now evaluate the accuracy vs toxicity bias for our concept regularization and leave-one-out PCA. Figure 5-14 compares the validation accuracy of our LR model vs PLDB. The red dot is standard logistic regression, with no fairness regularization. Green is the concept-regularized logistic regression described in the earlier sections. Blue are the results of leave-one-out PCA logistic regression, with the number denoting the n-th largest singular vector left out. Lines are for comparison to original without fairness regularization. As one can see, there is no silver bullet for correcting toxicity in all three models. In all three models one can leave out the second principal component at the expense of accuracy. For National Origin, one can remove the 5th principal component to decrease PLDB but this also would increase PLDB for gender. For concept regularization, this is less of an issue as we can differently weigh terms corresponding to various demographic groups. However, in Figure 5-14 all regularization terms were equally weighed to show the natural difference in PLDB for different demographic groups. CR in the logistic regression model has very different impacts on PLDB for different demographic groups. This is further proof that for effective use of mitigation techniques, application specific knowledge is needed. One application might more concerned with gender bias and develop a model with a higher weight for the corresponding CR term.

We show that it is possible to search for and find components of the data that have large toxicity bias. With LOOPCA, one can find, start to understand, and mitigate toxicity bias without strong prior knowledge for the meaning of each component.

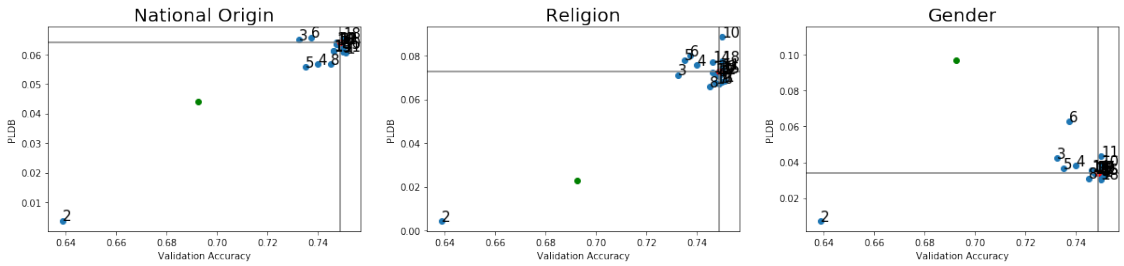


Figure 5-14: PLDB vs Accuracy. Red is standard logistic regression, with no fairness regularization. Green is the concept-regularized logistic regression described in the earlier sections. Blue are the results of leave-one-out PCA logistic regression, with the number denoting the n-th largest singular vector left out. Lines are for comparison to original without changes to the learning algorithm.

While combining LOOPCA and CR often leads to large decreases in model accuracy, we note general tradeoffs between the two techniques. With exception of the second principal component, LOOPCA had much more conservative results, perturbing accuracy and PLDB in small quantities. Meanwhile, concept regularization had a larger adverse effect on accuracy, though it did a better job of mitigating toxicity bias.

Furthermore, we also noticed that there is a large difference between performances for the three protected groups, throughout our results. Some groups, such as religion, seem to naturally have a much higher PLDB than others. Correspondingly, regularization appears to have a significantly different impact on PLDB with respect to each protected group, having a large impact on national origin, and a smaller impact on religion. This hints that finding a solution to prevent discrimination for all protected groups is perhaps more difficult than we thought, as the signals that compose each of these sensitive attributes are deeply ingrained in the dataset and embeddings in different ways.

5.3 Summary

In this chapter, we present results for three pieces of work.

- We showed that our Relative Negative Sentiment Bias framework can effectively measure unintended demographic bias in word embeddings. We also show how

this framework increases interpretabiliy for analyzing unintended bias for classic applications in NLP like sentiment analysis.

- We also showed that we can use adversarial learning to deccorelate word vectors with various concepts in language. For both toxicity and sentiment, we can effectively remove unintended demographic bias within word embeddings, resulting in fairer realistic downstream ML decision systems.

- Finally, we explored how we can use regularization and PCA at the learning algorithm level to mitigate unintended demographic bias. With respect to toxicity bias, our results for Concept Regularization and Leave-One-Out-PCA underscore the difficulty of mitigating toxicity bias with respect to multiple demographic groups at the same time. However, our mitigation techniques give practitioners the tools to tune PLDB/accuracy for their specific application.

At a higher level, we saw very promising results in our adversarial algorithm for debiasing word embeddings. This is a tool that can immediately have an impact in industry with limited fine tuning. At the later stage of mitigation at the learning algorithm level, the results were more intricate. As the learning algorithm contains uninteded demographic bias from multiple stage of the ML pipeline (embeddings dataset, and learning algorithm) more care must be taken by the practitioner to ensure that the right biases are mitigated for their application.

Chapter 6

Discussion and Future Work

In this section we summarize this thesis, emphasize major take-away messages, and highlight paths for future work.

6.1 Discussion

In this thesis we introduced the problem of fairness in machine learning and specialized down to understanding and mitigating unintended demographic bias for NLP applications like sentiment analysis and toxicity prediction. We now discuss various highlights and limitations of our work.

RNSB Framework for Measuring Unintended Demographic Bias in Word Embeddings

We showed that our RNSB framework can effectively measure unintended demographic bias in word embeddings with respect to negative sentiment. We can easily adapt this technique to any situation for which we wish to measure inequality in prediction for text. This allows us to measure inequality in positive sentiment or even toxicity by only changing the objective of the RNSB logistic regression algorithm. In the case of toxicity, one can substitute the positive and negative lexicon used to train the LR algorithm with a lexicon containing toxic and non toxic words. This permits the flexibility to evaluate risks for discrimination in downstream algorithms in many

scenarios.

It is important to note that we limited our framework to demographic identity keywords. However, demographic identity terms do not capture the whole story because they do not measure how using demographic identity terms in context can lead to unfair results. Future work should focus on mitigating inequality in sentiment or toxicity for sentences containing various protected attributes.

Mitigating Unintended Demographic Bias in Word Embeddings

In this work, we discussed removing sentiment and toxicity bias at the word embedding level. In the field of NLP, textual embeddings make up a substantial piece of machine learning pipelines, but bias can still enter from other sources. Because embeddings sit at the beginning of the ML pipeline, it is not possible to mitigate bias that enters at a later stage. Other techniques like bias mitigation at the learning algorithm level must be applied to more robustly tackle this problem. Furthermore, sentiment and toxicity bias can also enter into text via more abstract concepts than just single words. Phrases or even the author of the text may encode unintended demographic bias. Mitigating other sources of bias are part of future work.

Our results showed that we cannot perfectly decorrelate word vectors with sentiment no matter how largely we weigh our adversarial objective. This is most likely due to the fact that our debiasing model is linear. We did try some nonlinear models like encoder-decoder neural networks, but we found that the nonlinear mapping distorted the word vectors extensively. The word embedding vector space seems to be fragile, so there is a limited amount of debiasing one can do at this level without distorting too much of the word vector's meaning. On the other hand, because we used a linear model that directly minimized the movement of a word vector when debiasing, we were able to retain the meaning of the word vectors with respect to the embedding space.

Finally, there may be demographic identity terms that one may only want to debias in certain contexts. For example, *White*, could describe a crayon color or a demographic group. We leave it to the practitioner to use our technique to debias

word vectors in contexts where they are used as demographic attributes.

Mitigating Unintended Bias at the Learning Algorithm Level

Mitigating unintended demographic bias at the learning algorithm level allows one greater interpretabillity into machine learning itself. In this thesis, we developed a mitigation technique called Concept Regularization and applied it to a realistic toxicity prediction system. Though Concept Regularization caused our logistic regression algorithm to contain less unintended bias, regular L2 regularization also had a positive effect on the fairness in the algorithm. This hints at a more subtle aspect of ML and fairness. Often times, algorithmic unfairness can be attributed to overfitting to a part of the training set. Regularization techniques like CR and L2 regularization helps bias the ML solution towards simpler models not only increasing test accuracy, but also preventing models that work well only for the majority demographics in a dataset.

Applications Domains

In this thesis, we spend a lot of time breaking down fairness and unintended bias into definitions suited to NLP domains like sentiment analysis and toxicity prediction. However, we can uncover even more specific definitions. Within sentiment analysis, there are applications in predicting different types of emotion (anger, happiness, sadness) as regression, ordinal classifications problems, etc. In toxicity prediction, there are plenty of more specific problems like predicting offensiveness, racism, etc. In the interest of studying and debiasing practical applications, it is important to target more focused issues in machine learning.

6.2 Future Work

There is a lot of work to be done in both academia and industry to curb the unfairness in machine learning algorithms. Although there has been an abundance of work over the past few years into this area, there seems to be a gap between the mitigation tech-

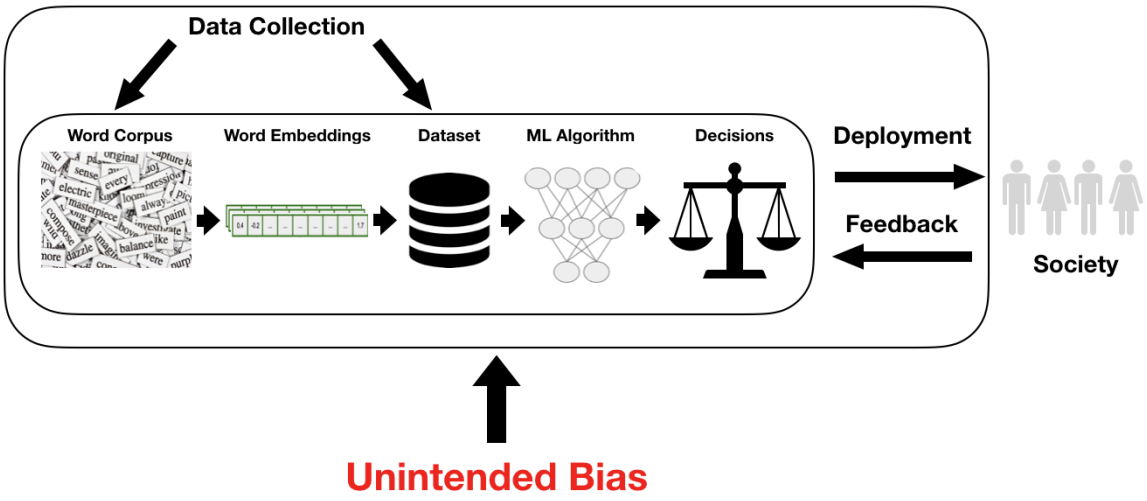


Figure 6-1: Typical workflow for a machine learning algorithm to be deployed into society. To really mitigate unintended demographic bias, it is important to understand the larger scope for sources of unfair biases.

niques and fairness metrics developed in academia and their use in practice. Fairness is a difficult concept to define both socially and technically, making it difficult to take abstract solutions from academia and apply them in practice. Thus, it is essential that researchers target more specific problems in machine learning and its applications. In this work we took inspiration from many general techniques for mitigating unintended bias and created techniques tailored to specific real world applications in NLP. We took a look at many sources of unintended demographic bias within a typical NLP pipeline in ML. However, there are more sources of unintended bias that arise when considering how ML applications are developed in practice. Figure 6-1 illustrates how unintended bias can enter the development of machine learning systems through *data collection*, *deployment*, and *feedback*.

- *Data Collection*: It is important not to consider a dataset as given, but understand what choices where made to collect data. What purpose was the data collected for? Are we fairly collecting samples? Many of the imbalances in datasets come from issues in the data collection process.

- *Deployment*: The way a ML algorithm is used in practice could also be a source of bias that causes unfairness. Is the ML algorithm being applied to domain

that differs from the dataset collection? Questions like this can help catch unfair applications of ML.

- *Feedback*: As society is ephemeral, it is important to have channels of feedback to help ML models evolve, or catch mistakes that cause discrimination. Many ML systems fail to have the appropriate channels of feedback, resulting in a lack of interpretability and unfairness.

As fairness in machine learning is a real world problem, it is important to put our machine learning pipeline in the context of how it is deployed in practice. Future work should continue to understand and mitigate unintended bias in real world machine learning systems.

Bibliography

[1] Eneko Agirre, Enrique Alfonseca, Keith Hall, Jana Kravalova, Marius Paşca, and Aitor Soroa. A study on similarity and relatedness using distributional and wordnet-based approaches. In *NAACL*, pages 19–27, 2009.

[2] Sebastian Benthall and Bruce D. Haynes. Racial categories in machine learning. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 289–298, New York, NY, USA, 2019. ACM.

[3] Alex Beutel, Jilin Chen, Zhe Zhao, and Ed H Chi. Data decisions and theoretical implications when adversarially learning fair representations. *FATML*, 2017.

[4] Su Lin Blodgett and Brendan O'Connor. Racial disparity in natural language processing: A case study of social media african-american english. *arXiv preprint arXiv:1707.00061*, 2017.

[5] Tolga Bolukbasi, Kai-Wei Chang, James Zou, Venkatesh Saligrama, and Adam Kalai. Quantifying and reducing stereotypes in word embeddings. *ICML*, 2016.

[6] Tolga Bolukbasi, Kai-Wei Chang, James Y Zou, Venkatesh Saligrama, and Adam T Kalai. Man is to computer programmer as woman is to homemaker? debiasing word embeddings. In *NIPS*, pages 4349–4357, 2016.

[7] Aylin Caliskan, Joanna J. Bryson, and Arvind Narayanan. Semantics derived automatically from language corpora contain human-like biases. *Science*, 356(6334):183–186, 2017.

[8] Jiahao Chen, Nathan Kallus, Xiaojie Mao, Geoffry Svacha, and Madeleine Udell. Fairness under unawareness: Assessing disparity when protected class is unobserved. In *Proceedings of the Conference on Fairness, Accountability, and Transparency*, FAT* '19, pages 339–348, New York, NY, USA, 2019. ACM.

[9] Alexandra Chouldechova. Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. *Big data*, 5(2):153–163, 2017.

[10] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain, and Lucy Vasserman. Measuring and mitigating unintended bias in text classification. *AAAI*, 2018.

[11] Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. Fairness through awareness. pages 214–226. ACM, 2012.

[12] Harrison Edwards and Amos Storkey. Censoring representations with an adversary. *ICLR*, 2016.

[13] Yanai Elazar and Yoav Goldberg. Adversarial removal of demographic attributes from text data. *arXiv preprint arXiv:1808.06640*, 2018.

[14] Sahaj Garg, Vincent Perot, Nicole Limtiaco, Ankur Taly, Ed H Chi, and Alex Beutel. Counterfactual fairness in text classification through robustness. *arXiv preprint arXiv:1809.10610*, 2018.

[15] Moritz Hardt, Eric Price, Nati Srebro, et al. Equality of opportunity in supervised learning. In *NIPS*, pages 3315–3323, 2016.

[16] Tatsunori B Hashimoto, Megha Srivastava, Hongseok Namkoong, and Percy Liang. Fairness without demographics in repeated loss minimization. *ICML*, 2018.

[17] Peter Henderson, Koustuv Sinha, Nicolas Angelard-Gontier, Nan Rosemary Ke, Genevieve Fried, Ryan Lowe, and Joelle Pineau. Ethical challenges in data-driven dialogue systems. *CoRR*, 2017.

[18] Dirk Hovy and Shannon L. Spruit. The social impact of natural language processing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 591–598. Association for Computational Linguistics, 2016.

[19] Minqing Hu and Bing Liu. Mining and summarizing customer reviews. In *ACM*, pages 168–177, 2004.

[20] Mohit Iyyer, Peter Enns, Jordan Boyd-Graber, and Philip Resnik. Political ideology detection using recursive neural networks. In *ACL*, volume 1, pages 1113–1122, 2014.

[21] S. Mattu J. Angwin, J. Larson and L. Kirchner. Machine bias: Theres software used across the country to predict future criminals. and its biased against blacks. 2016.

[22] Faisal Kamiran and Toon Calders. Data preprocessing techniques for classification without discrimination. *Knowledge and Information Systems*, 33(1):1–33, Oct 2012.

[23] Toshihiro Kamishima, Shotaro Akaho, Hideki Asoh, and Jun Sakuma. Fairness-aware classifier with prejudice remover regularizer. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 35–50. Springer, 2012.

[24] Svetlana Kiritchenko and Saif M Mohammad. Examining gender and race bias in two hundred sentiment analysis systems. *Proceedings of the 7thJoint Conference on Lexical and Computational Se-mantics(*SEM), New Orleans, USA.*, 2018.

[25] Jon M. Kleinberg, Sendhil Mullainathan, and Manish Raghavan. Inherent trade-offs in the fair determination of risk scores. *ITCS*.

[26] Bing Liu. Sentiment analysis and opinion mining. *Synthesis lectures on human language technologies*, 5(1):1–167, 2012.

[27] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[28] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *NIPS*, pages 3111–3119, 2013.

[29] Saif Mohammad, Felipe Bravo-Marquez, Mohammad Salameh, and Svetlana Kiritchenko. Semeval-2018 task 1: Affect in tweets. In *Proceedings of The 12th International Workshop on Semantic Evaluation*, pages 1–17. Association for Computational Linguistics, 2018.

[30] Maryam Najafian, Wei-Ning Hsu, Ahmed Ali, and James Glass. Automatic speech recognition of arabic multi-genre broadcast media. In *ASRU*, pages 353–359, 2017.

[31] Maryam Najafian, Saeid Safavi, John HL Hansen, and Martin Russell. Improving speech recognition using limited accent diverse british english training data with deep neural networks. In *MLSP*, pages 1–6, 2016.

[32] C.L. Blake D.J. Newman and C.J. Merz. UCI repository of machine learning databases, 1998.

[33] Ben Packer, Yoni Halpern, Mario Guajardo-Cspedes, and Margaret Mitchell. Text embedding models contain bias. here's why that matters. Google Developers, 2018.

[34] Bo Pang, Lillian Lee, et al. Opinion mining and sentiment analysis. *FTIR*, 2(1–2):1–135, 2008.

[35] Ji Ho Park, Jamin Shin, and Pascale Fung. Reducing gender bias in abusive language detection. *EMNLP*, 2018.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *EMNLP*, pages 1532–1543, 2014.

[37] Adrián Pérez-Suay, Valero Laparra, Gonzalo Mateo-García, Jordi Muñoz-Marí, Luis Gómez-Chova, and Gustau Camps-Valls. Fair kernel learning. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, pages 339–355. Springer, 2017.

[38] Marta Recasens, Cristian Danescu-Niculescu-Mizil, and Dan Jurafsky. Linguistic models for analyzing and detecting biased language. In *ACL*, volume 1, pages 1650–1659, 2013.

[39] Prasanna Sattigeri, Samuel C Hoffman, Vijil Chenthamarakshan, and Kush R Varshney. Fairness gan. *arXiv preprint arXiv:1805.09910*, 2018.

[40] Sucheta Soundarajan and Daniel Clausen. Equal protection under the algorithm: A legal-inspired framework for identifying discrimination in machine learning. *FATML*, 2018.

[41] Robert Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *AAAI*, pages 4444–4451, 2017.

[42] Robyn Speer. Conceptnet numberbatch 17.04: better, less-stereotyped word vectors. ConceptNet, 2017.

[43] Rachael Tatman. Gender and dialect bias in youtube's automatic captions. In *Proceedings of the First ACL Workshop on Ethics in Natural Language Processing*, pages 53–59, 2017.

[44] Theresa Wilson, Janyce Wiebe, and Paul Hoffmann. Recognizing contextual polarity in phrase-level sentiment analysis. In *EMNLP*, pages 347–354, 2005.

[45] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. Ex machina: Personal attacks seen at scale. In *Proceedings of the 26th International Conference on World Wide Web*, pages 1391–1399. International World Wide Web Conferences Steering Committee, 2017.

[46] Muhammad Bilal Zafar, Isabel Valera, Manuel Gomez Rodriguez, and Krishna P Gummadi. Fairness constraints: Mechanisms for fair classification. *PMLR*, 2015.

[47] Rich Zemel, Yu Wu, Kevin Swersky, Toni Pitassi, and Cynthia Dwork. Learning fair representations. In *ICML*, volume 28, pages 325–333, 2013.

[48] Brian Hu Zhang, Blake Lemoine, and Margaret Mitchell. Mitigating unwanted biases with adversarial learning. *AIES*, 2018.

[49] Jieyu Zhao, Tianlu Wang, Mark Yatskar, Vicente Ordonez, and Kai-Wei Chang. Men also like shopping: Reducing gender bias amplification using corpus-level constraints. *EMNLP*, 2017.