Dbias: Detecting biases and ensuring Fairness in news articles

Shaina Raza[1*], Deepak John Reji[2], Chen Ding[3]

Abstract

The problem of fairness is garnering a lot of interest in the academic and broader literature due to the increasing use of data-centric systems and algorithms in machine learning. This paper introduces Dbias (https://pypi.org/project/Dbias/), an open-source Python package for ensuring fairness in news articles. Dbias can take any text to determine if it is biased. Then, it detects biased words in the text, masks them, and suggests a set of sentences with new words that are bias-free or at least less biased. We conduct extensive experiments to assess the performance of Dbias. To see how well our approach works, we compare it to the existing fairness models. We also test the individual components of Dbias to see how effective they were. The experimental results show that Dbias outperforms all the baselines in terms of accuracy and fairness. We make this package (Dbias) as publicly available for the developers and practitioners to mitigate biases in textual data (such as news articles), as well as to encourage extension of this work.

Keywords: *Bias; fairness; Transformer-based models; deep learning; entity recognition; classification; masking*

1 Introduction

Natural language processing (NLP) is a branch of artificial intelligence (AI) that assists computers in understanding, interpreting, and manipulating human language [1]. NLP problems can be solved with the help of Machine Learning (ML), which can automate processes and deliver accurate responses [2]. In recent years, the deep learning techniques have demonstrated promising results in narrowing down the gap between sequence-to-sequence learning approaches and human-like performance in a variety of NLP tasks [3]. One common thing about ML models is that they are often trained on a large text corpus (e.g., Google BERT, Facebook BART and so other industry driven products), which may introduce substantial biases into the models. These biases are usually passed on to the models during the training time [4, 5], and often the developers are unaware of these biases [6].

✉ Shaina Raza Shaina.raza@utoronto.ca
Deepak John Reji deepak.reji@erm.com
Chen Ding cding@ryerson.ca

[1] University of Toronto, Ontario, Canada
[2] Environmental Resources Management, Bangalore, India
[3] Ryerson University, Toronto, Ontario, Canada

There are numerous examples of biases in ML models due to data, such as Amazon's hiring algorithm [7] that favored men, Facebook targeted housing advertising that discriminated on the basis of race and color [8], and a healthcare algorithm [9] that exhibited significant racial biases in its recommendations [10]. Data-heavy systems, such as news recommender systems, are also trained on huge volumes of data with limited control over the quality of the training data [11]. These news recommenders often inherit biases from the data [12], potentially influencing the beliefs and behaviors of news consumers [13]. To mitigate biases, the biased models are frequently eliminated, as was the case with Amazon's hiring algorithm [7], which is not always a feasible solution. As discussed in the literature [5, 14, 15], it is necessary to eliminate these biases early in the data collection process, before they enter the system and are reinforced by model predictions, resulting in biases in the model's decisions. In this study, we aim to eliminate biases in the original data as soon as possible, such as during the data ingestion time.

Bias detection and mitigation are hot topics both in academia and industry. Fairness and bias have been defined in a variety of ways by the academic researchers. Traditionally, the bias is defined as prejudice in favour of or against a particular thing, person, or group in comparison to another, usually in an unjust manner [6, 16].

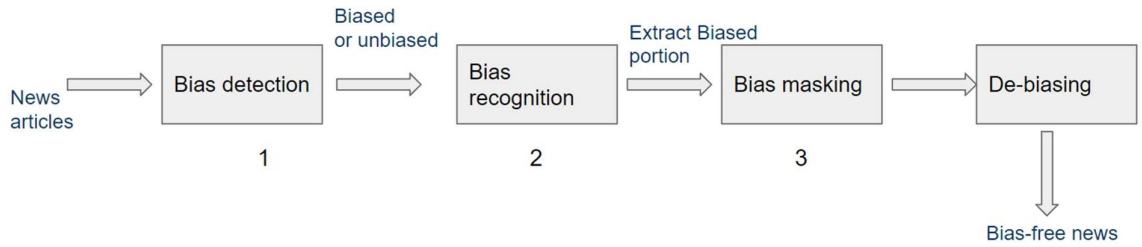

Figure 1: Debias – a fair ML pipeline for news articles

For example, gender, race, demographic or sexual orientation are some of the examples of the biases [5]. The objective of fairness is to identify and reduce the impacts of various biases [17]. The goal of fiarness is to prepare these ML systems to avoid perpetuating human and societal biases or adding new biases into the context.

Biases in the ML models can be mitigated at three stages: early, mid, and late [17]. The early stage would be to reduce bias by manipulating the training data before training the algorithm [10, 18, 19]. The mid stage would be to de-bias the model itself [4, 19–21], which is also framed as an optimization problem [22]. The late stage refers to reducing the biases by manipulating the output predictions after training the algorithm [23, 24]. Prior research [25–27] has demonstrated that missing the chance to detect bias at an early stage of the ML pipeline might make algorithmic fairness (mid or late stages of bias mitigation) difficult to achieve. In this paper, we propose a fair ML pipeline that takes raw data and de-biases it early on, ensuring fairness throughout the pipeline's phases.

A ML pipeline is composed of several steps that facilitate the automation of ML workflows [28]. Many real-world applications, for example, the Netflix recommender system [29], Spark healthcare [30] and Uber forecasting [28] are often represented as ML pipelines. Usually, these pipelines are designed to input some data as features and generates a score that predicts, classifies or recommends future outcomes. In contrast to typical product-driven ML pipelines, we propose a fair ML pipeline, designed exclusively for mitigating data biases and leveraging fairness in applications. We develop, Dbias – a fair ML pipeline, shown in Figure 1, which mitigates biases in the data and propagates fairness through different steps (data preprocessing, model training, analysis and development) of the pipeline.

As shown in Figure 1, Dbias first detects biases in the text, then it recognizes the biased words, masks and replace those biased words with new (non-biased or at least less biased) words to de-bias the text. Each of these phases in Dbias pipeline consists of its own training and inference steps (shown in Figure 2).

To the best of our knowledge, there is no ML pipeline developed exclusively to mitigate biases in data. The majority of research on fairness in ML has focused on classification tasks [16, 31–33]. By focusing exclusively on the fairness of classifiers (e.g., Decision Tree, Logistic Regression), we miss the impact of fairness on other stages in a standard ML pipeline.

FairML [34], FairTest [33], Themis-ml [35] and AIF360 [6] are some of the well-known ML fair pipelines that implement and evaluate the state-of-the-art fairness algorithms. This means that these pipelines use and evaluate off-the-shelf fair ML models [16, 31–33, 36, 37]. In comparison to these approaches (fairness methods and pipelines), Dbias is a self-contained fair ML pipeline that has its own algorithms for detecting and mitigating biases. Dbias ensures that the fairness is maintained throughout the pipeline.

The Dbias can detect biases in any kind of text. The only requisite is to train the model on the domain-specific data. However, in this paper, we will focus on news media bias. According to research [11, 38, 39] news media can be biased, and this biased coverage has the potential to significantly influence public perceptions of the reported topics. Biased news media can also result in "filter bubbles" or "echo chambers" [11, 40, 41], which can lead to a lack of understanding of specific concerns as well as a narrow and one-sided point of view. This inspires us to train Dbias to mitigate news media biases.

Contributions: We summarize our contributions as:

1. We develop a fair ML pipeline, which we name as Dbias (de-biasing) that de-bias the text (e.g., news text). To make it easier for practitioners to use, we follow the widely acknowledged data science ML pipeline structure [42] to build Dbias. We develop and package different algorithms for bias detection,

recognition, masking and de-biasing into the Dbias pipeline.

2. We make Dbias available as an open-source package distributed under the MIT[1] License. It is publicly shared in the GitHub repository[2] and as the PyPi project[3]. The released package also includes introductory tutorials to the concepts, as well as documentation, usage and guidance to assist data scientists and practitioners in incorporating this package into their work products.
3. Even though Dbias is released as a generic fair ML pipeline, it can be applied to any other text data. However, focusing on news media biases in this work, we demonstrate in our GitHub tutorials how we can use Dbias in conjunction with API code blocks (such as those found in Google News API) to reduce biases in news articles.
4. We conducted extensive experiments to compare Dbias' performance to that of cutting-edge fairness methods. We also put the individual components of Dbias through their paces to see how well they perform in various bias mitigation tasks. Furthermore, we demonstrate through our experiments that there exists a trade-off between accuracy and fairness, which also previous research [6, 10, 43].

The rest of the paper is organized as follows: Section 2 is the related work, section 3 is the working of the Dbias architecture, section 4 is about the experimental setup, section 5 is the discussion about results and analysis, section 6 is the discussion section, and finally section 7 is the conclusion.

2 Literature Review

As AI is increasingly used in highly sensitive domains such as news, health care, hiring, journalism, and criminal justice, there is a growing awareness of the consequences of embedded biases and unfairness. Numerous studies have shown that AI is capable of embedding and deploying human and societal biases into the solutions [5, 6, 23]. For example, the Correctional Offender Management Profiling for Alternative Sanctions (COMPAS) algorithm mislabeled African-American defendants nearly twice as often as white defendants [44]. The inability of ML models to mitigate these undesirable biases is a significant impediment to AI reaching its full potential.

Fairness [17] is a multi-faceted concept that varies by culture and context. It is quite difficult to have a standard definition of fairness as each definition depends on a different use case and organization. Distinct definitions of fairness can lead to different outcomes [45]. There exists at least 21 mathematical definitions for fairness in politics [46]. A decision tree[4] on different definitions of fairness is provided by the University of Chicago. Overall, it is quite difficult to meet many definitions of fairness at the same time [14].

Most of the definitions on fairness focus on either individual fairness (treating similar individuals fairly) [17, 37] or group fairness (equitably distributing the model's predictions or outcomes across groups) [17, 47]. Individual fairness aims to ensure that statistical measures of outcomes are equal for people who are statistically similar. Group fairness divides a population into distinct groups based on protected characteristics, with the goal of ensuring that statistical measures of outcomes are comparable across groups.

Bias is defined as an inclination or prejudice for or against one person or group, especially in an unfair manner [5]. While algorithmic bias is frequently discussed in the literature of ML, in most situations it is the underlying data that introduces bias. For example, if there had been an equal amount of data for men and women in Amazon's recruiting[7] algorithm, the algorithm might not have biased as much.

Fairness algorithms: In the research of AI and ML fairness [6, 17], the bias mitigation algorithms are categorized into three broad types: (1) pre-processing algorithms; (2) in-processing algorithms and (3) post-processing algorithms. These algorithms are briefly discussed below.

Pre-processing algorithms: The pre-processing algorithms attempt to learn a new representation of data by removing the information associated with the sensitive attribute, while retaining as much of the actual data as possible. This technique manipulates the training data prior to training the algorithm. Well-known pre-processing algorithms are:

- A reweighting [10] algorithm generates weights for the training samples in each (group, label), without changing the actual feature or label values.
- The learning fair representations [19] algorithm discovers a latent representation that encodes the data, while concealing information about protected attributes. Protected attributes are those attributes

[1] https://opensource.org/licenses/MIT
[2] https://github.com/dreji18/Fairness-in-AI
[3] https://pypi.org/project/Dbias/
[4] http://aequitas.dssg.io/static/images/metrictree.png

that divide a population into groups whose outcomes should be comparable (such as race, gender, caste, and religion) [17].
- The disparate impact remover [31] algorithm modifies feature values to improve group fairness, while keeping rank ordering within groups.
- Optimized pre-processing [48] algorithm learns a probabilistic transformation that edits data features and labels for individual and group fairness.

Usually, pre-processing techniques are easy to use as the modified data can be used for any downstream tasks without requiring any changes to the model itself.

In-processing algorithms: In-processing algorithms penalize the undesired biases from the model, to incorporate fairness into the model. The in-processing technique influences the loss function during the model training to mitigate biases. In the past, the in-processing algorithms [49, 50] have been used to provide equal access to racially and ethnically diverse group. Some of the example in-processing algorithms are listed below:
- Prejudice remover [49] augments the learning objective with a discrimination-aware regularization term.
- Adversarial De-biasing [4] algorithm learns a classifier to maximize the prediction accuracy while decreasing an adversary's ability to deduce the protected attribute from the predictions.
- Exponentiated gradient reduction [51] algorithm breaks down fair classification into a series of cost-sensitive classification problems, returning a randomized classifier with the lowest empirical error subject to fair classification constraints.
- Meta fair classifier [50] algorithm inputs the fairness metric and returns a classifier that is optimized for the metric.

The in-processing technique is model or task-specific, and it requires changes within the algorithm, which is not always a feasible option.

Post-processing algorithms: The post-processing algorithms manipulate output predictions after training to reduce bias. These algorithms can be applied without retraining the existing classifiers (as in in-processing). Some of the post-processing algorithms are:
- Reject option classification [36] algorithm gives favorable outcomes (labels that provide advantage to an individual or group, e.g., being hired for a job, not fired) to unprivileged groups.
- Equalized odds [24] algorithm changes the output labels to optimize equalized odds through linear programming.
- Calibrated equalized [23] odds optimizes score outputs to find probabilities with which to change output labels with an equalized odds objective.

Usually, the post-processing technique requires access to protected attributes late in the pipeline.

Recently, the software engineering community has also started to work on fairness in ML, specifically fairness testing [33, 52]. Some work has been done to develop automated tools, such as AI Fairness 360 [6], FairML [34] , Aequitas [53], Themis-ML [35], FairTest [33], that follows a software development lifecycle.

Transfer learning techniques: Transfer learning is a technique to transfer the knowledge contained in larger, different but related source domain to a target domain [54]. The goal is to improve the performance of target domain with the existing knowledge of the source domain. Bidirectional Encoder Representations from Transformers (BERT) [55] is an example, which has shown state-of-the-art performance in many tasks like classification, question-answering and so.

We discuss the state-of-the-art transfer learning methods used in the bias mitigation and fairness research. Li et al. 2021 [56] study the gender bias inside the Transformer-based model (BERT). They [56] calculate the attention scores for the corresponding gender pronouns and occupations, swap the gender pronouns to eliminate the position effect on bias judgement, and then again check the consistency of the gender bias associated with the occupation. Sinha and Dasgupta [57] employ the BERT model to detect biases from the text. Both these models, while equally important, are primarily concerned with the detection and extraction of biased sentences.

The task of identifying a named entity (a real-world object or concept) in unstructured text and then classifying the entity into a standard category is known as named entity recognition [58]. Mehrabi et al. 2020 [32] used named entities to determine whether female names are more frequently tagged as non-person than male names. Some other researchers [59, 60] use the named entities to identify biases based on occupation, race, and demographics. These named entity recognition models usually recognize the biased entities from the data, without mitigating them.

The masked language modeling is also used to identify biases. Based on two crowdsourced datasets, Kaneko and Bollegala 2021 [61] propose a technique to accurately predict different types of social biases in text. They [61] demonstrate that social biases do exist in masked language models and suggest developing

methods to robustly debias pre-trained masked language models as a future direction.

Fairness toolkits: We discuss fairness toolkits here:

- FairML [34] is a toolkit that uses few ranking algorithms to quantify the relative effects of various inputs on a model's predictions, which can be used to assess the fairness in models.
- FairTest [33] is a python package that learns a special decision tree that divides a user population into smaller subgroups with the greatest possible association between protected (such as gender, race, and other characteristics deemed sensitive features) and algorithm outputs.
- Themis-ml [35] is a python library that implements several state-of-the-art fairness-aware methods that comply with the sklearn API.
- AIF360 [6] consists of a number of fairness metrics for datasets and state-of-the-art ML models to mitigate biases in the datasets and models.

Fairness metrics*:* To ensure the fairness, the ML community has also proposed various fairness metrics. For example, the disparate impact ratio compares the rate at which an underprivileged groups (groups having systematic biases e.g., females) receives a particular outcome compared to a privileged group (groups with systematic advantages, e.g., males) [31]. In some works [17], the number of positives and negatives for each individual or group, difference of means, odds ratio are also computed to measure fairness.

Comparison with the state-of-the-art approaches: While each of the previous works discussed in this section is valuable and incremental, they are primarily concerned with fairness across different tasks (pre-processing, in-processing, and post-processing). These models either remove biases and ensure fairness either during pre-processing, in-processing or post-processing. The fairness toolkits (AIF360, FairML, FairTest, etc.) also take the existing approaches to mitigate the biases in different groups (gender, populations, religions).

In this work we combine the strength of deep neural networks and Transformer-based architecture into our fair ML pipeline. We propose that fairness can be achieved by ensuring that each component of the ML pipeline is fair. We do not rely on existing built-in fairness models to ensure that data or model is fair; rather, we construct a fair ML pipeline comprised of multiple components that takes raw data and de-biases it. Also, compared to other methods [32, 56, 60, 61], which either detect, recognize or only mask the biased words/sentences, we consider all these aspects of bias detection and mitigation in a single pipeline. Different from the previous works, we do not only focus on the specific biases (gender, race, ethnicity), infact, we develop the Dbias to detect and mitigate many kinds of biases from the text data. The only requisite is to fine-tune the model on the domain specific data.

3 DBias, a fair ML pipeline

In this section, we define the preliminaries, problem definition, overview, and underlying working of Dbias.

3.1 Preliminaries

We use the following key terms [5, 17] in this work.

- *Bias* is a type of systemic error.
- A *protected attribute* divides a population into groups, for example, race, gender, caste, and religion.
- A *privileged* value of a protected attribute denotes a group that historically has a systematic advantage, for example, male gender, white race.
- An *underprivileged* group faces prejudice based on systematic biases such as gender, race, age, disability, and so on.
- *Group fairness* means that the protected attribute receives similar treatments or outcomes.
- *Individual fairness* means that similar individuals receive similar treatments or outcomes.
- *Equalized odds* [62] is a statistical notion of fairness that ensures classification algorithms do not discriminate against protected groups.
- *Fairness* or *fair* is a broad term used in ML and generally refers to the process of undoing the effects of various biases in the data and algorithms.

The goal of *fairness* is to mitigate the unwanted biases that benefits privileged groups and disadvantages unprivileged.

Table 1: Some biases in news domain

Bias	Example
Gender	All *man* hours in *his* area of responsibility must be approved.
Age	Apply if you are a *recent* graduate.
Racial/ Ethnicity	Police are looking for any *black* males who may be involved in this case.
Disability	Genuine concern for the *elderly* and *handicapped.*
Mental health	Any experience working with *retarded* people is required for this job.
Other biases addressed: Religion, education, political ideology (liberal, conservative)	

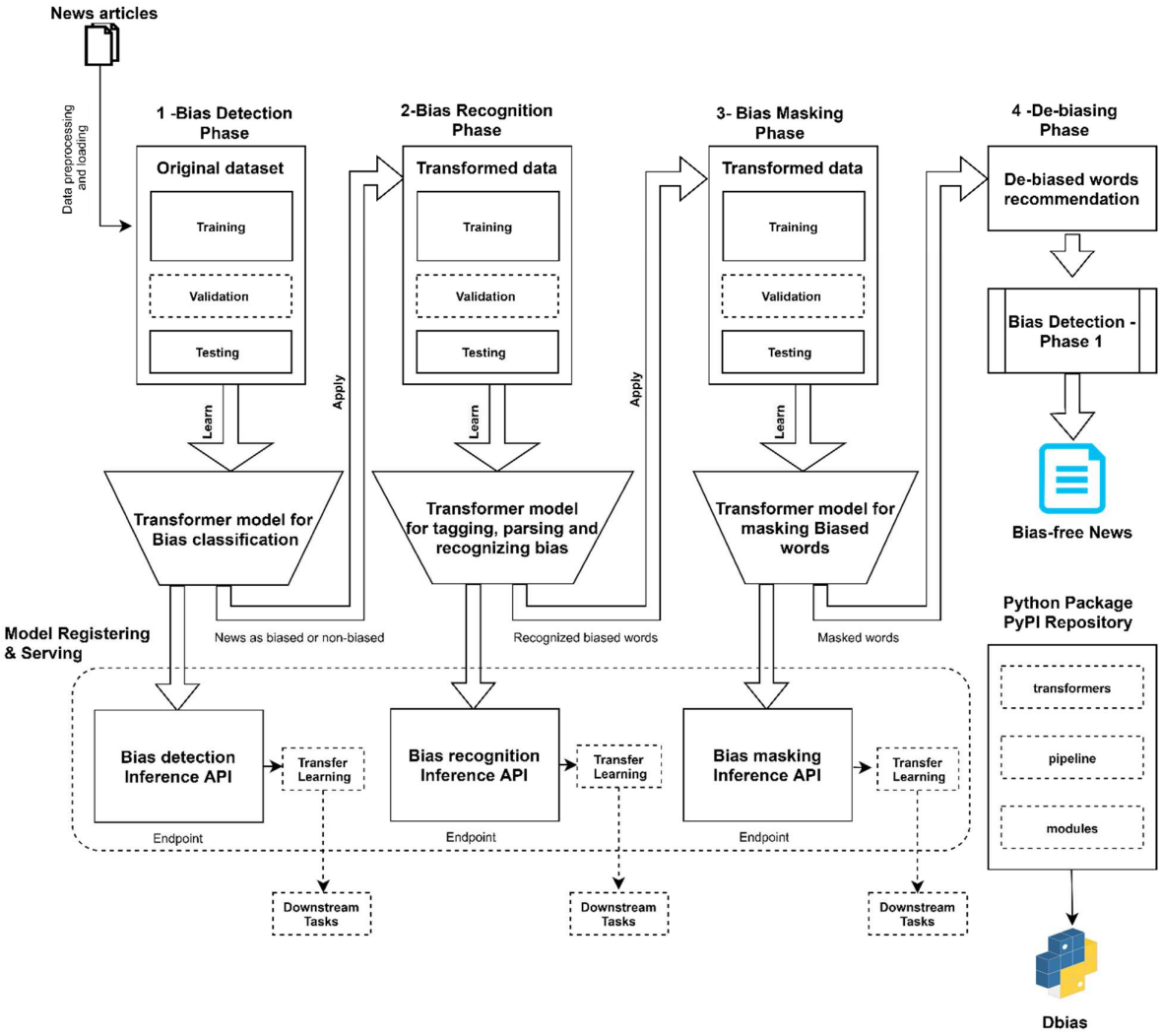

Figure 2: Dbias and its debiasing workflow

In Table 1, we define some of the biases in news media [38, 63] that we try to mitigate in this work. However, depending on the data on which our model is fine-tuned, we can also address many other types of biases across other domains.

3.2 Problem definition

Given a set of news articles that may contain a variety of biases, the task is to detect, recognize and undo those biases from the data. The goal of this research is to mitigate unfairness in the news domain.

3.3 Overview of the Dbias

In this section, we present the workflow of the Dbias. Figure 2 shows Dbias, which is a fair ML pipeline. We often use the word fair ML pipeline, in this work, to refer to multiple sequential steps that perform steps from data extraction to preprocessing, data preprocessing to modeling, data transformation and deployment in the Dbias pipeline. The main phases in this pipeline are bias detection, bias recognition, bias masking and de-biasing. Each phase gets the input from the preceding phase, except for the first phase (bias detection phase) that gets the input from a news API (e.g., Google news). Each phase has a training, validation, and testing task. The transformation of the data from one phase to another phase is shown by a trapezoid.

Each model is served as an inference API and can function individually as well. The output from each trapezoid goes to the respective inference API. For example, the transformed data from the bias detection phase goes into the bias detection inference API. Once the models are trained and tested, each of them is registered and deployed so that it can be used as a callable API. The goal of the inference API is to execute inference queries over the new transformed data without the need of loading all the heavy weighted python libraries. The final output of the pipeline is the news articles (or any text) that are free

from biases. There are also various stages in the pipeline where we evaluate the modules using fairness metrics (not pictured). These will be discussed in the evaluation section.

Our goal is to enable end-users to shift seamlessly from raw biased data to a fair model. Next, we explain each phase of Dbias pipeline.

3.3.1 Data collection and preparation phase

We prepare the annotated dataset MBIC (Media Bias Including Characteristics) [38] that represents various bias instances in the news articles to train our models. More details about the dataset are given in Section 4.

3.3.2 Model training, fine-tuning, and testing

In this work, we fine-tune the state-of-the-art pretrained Transformer models, such as BERT (Devlin et al. 2018), DistilBERT (Sanh et al. 2019), RoBERTa (Liu et al. 2019) on our MBIC dataset for various downstream tasks. We use the training checkpoints of these models (BERT, DistilBERT and RoBERTa) and fine-tune them using MBIC dataset. For example, we fine-tune the DistilBERT model on the MBIC dataset for the bias classification task. Similarly, we fine-tune the RoBERTa on our MBIC dataset for the bias recognition task. We also adapt the BERT architecture for masking the biased words and filling in those masked words with the non-biased words. We discuss the details of each of such tasks in Section 3.4.

3.3.3 Model registering and sharing

We have registered and shared our models as TensorFlow checkpoints on the Huggingface.co website. We create the model cards for the tasks: bias detection[5] and recognition[6]. Model cards are the markdown files providing useful information for discoverability, reproducibility and sharing of the model. We also provide the inference (running live data points) API along with each model card. Our purpose is to make it easier for the other researchers and developers to use our model and to contribute.

3.3.4 Packaging

We group different modules of Dbias pipeline into a single package. Our whole package is hosted on http:/pypi.org under the MIT licence and can be installed using the command *pip install Dbias*.

Next, we discuss our methodology and cover the details about bias detection, recognition, and de-biasing steps.

3.4 Methodology

The specific tasks to perform in this work are:

- Bias detection: To detect whether a news article is biased or not.
- Bias recognition: To recognize the biased words or phrases from the news articles.
- De-biasing: To de-bias the data by replacing the biased words or phrases from the news article with unbiased or at least less biased word(s). The de-biasing also consists of masking, i.e., to hide the biased words.

3.4.1 Bias detection module

The input to the Dbias is a set news articles that potentially contain biased instances. The task of the bias detection module is to predict if a sentence is biased or non-biased. For instance, given the news headline, "Don't buy the pseudo-scientific hype about tornadoes and climate change" [64], the bias detection module should be able to detect the sentence's bias and classify it as biased news.

The Bias detection module is built upon the BERT (Devlin et al. 2018) architecture. BERT has achieved state-of-the-art results in a variety of NLP tasks, including text classification and natural language understanding, so we fine-tune it for our specific use case of bias detection. In our preliminary experiments, we fine-tuned and evaluated various Transformer-based models (e.g., BERT, GPT-2, and others from the Huggingface Transformers' library) on our dataset and discovered that the DistilBERT (Sanh et al. 2019) model achieves both higher accuracy and faster inference speed, which is why we chose to work with it. DistilBERT is a distilled (approximate, faster and smaller) version of BERT.

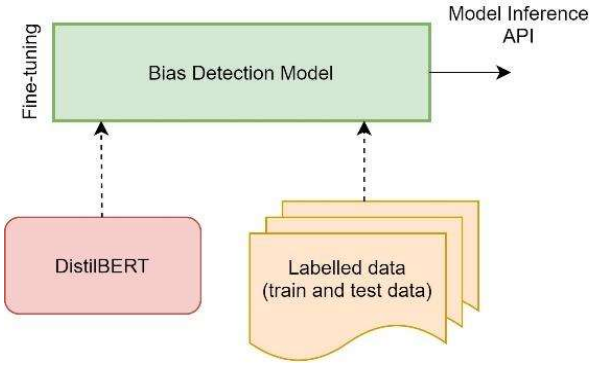

Figure 3: Bias detection module

[5] https://huggingface.co/d4data/en_pipeline

[6] https://huggingface.co/d4data/bias-detection-model

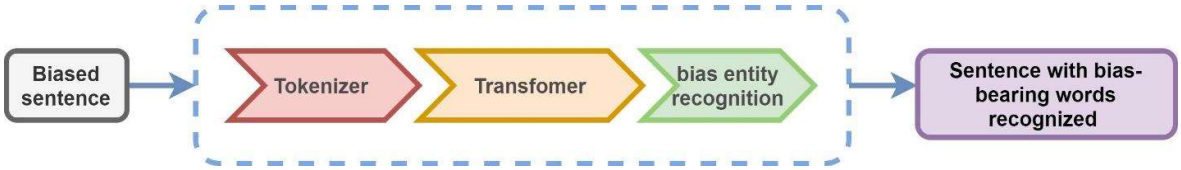

Figure 4: Bias recognition pipeline

We use the DistilBert (distilbert-base-uncased) with these details: Uncased: 6-layer, 768-hidden, 12-heads, 66M parameters, and we fine-tune it on MBIC dataset. For binary classification, we use binary-cross entropy loss [65] with sigmoid activation function. The output from the bias detection model is a set of news articles that are classified as biased or non-biased. Our bias detection module is shown in Figure 3.

3.4.2 Biased recognition module

The second module in the Dbias pipeline is the bias recognition module. The task of the bias recognition module is to annotate the biased words in the news articles with tags, indicating that each tagged word is biased. This module takes as input a set of news articles that have been identified as biased in the preceding module (bias detection), and outputs a set of news articles where the biased words are picked and recognized. The news headline "Don't buy the pseudo-scientific hype about tornadoes and climate change," for example, has already been classified as a biased sentence by the preceding bias detection module, and the biased recognition module can now identify the term "pseudo-scientific hype" as a biased word.

Traditionally, named entity recognition (NER) is the task of identifying a named entity (a real-world object or concept) in unstructured text and then classifying the entity into a standard category [58]. Though NER is not directly related to bias identification, the research [5, 59] shows that NER models can be used to examine the existence and level of biasness in data, which obviously helps in mitigating biases in various applications. For example, a NER model can be used to find if there are more female names tagged as non-person than male names [32] or to identify biases based on occupation, race, and demographics [59].

In our work, we take a unique approach to identifying biases in news articles. Rather than looking for conventional entities (Name, Location, Event, and Organization, which are primarily nouns), we look for bias-bearing words associated with each entity. Specifically, we refer to each entity in the NER task as an bias-bearing entity that is manifested in syntax, semantics or in the linguistic context (e.g., the word 'pseudo-scientific hype' is a biased word).

We show our bias recognition module, which is also a pipeline, in Figure 4:

We use a Transformer-based model in our bias recognition module. A standard NER [58] task has traditionally been viewed as a sequence labelling problem with word-level tags. The standard NER models are based on dictionaries or rules that may fail to recognize references to unknown entities in the text (for example, biased words as in our work). Therefore, we employ a Transformer-based model as a wrapper for the bias recognition task. According to recent research [66, 67], the Transformer-based model can also detect long-term context in data, allowing us to identify more bias-bearing words in the text.

We employ the RoBERTa (Liu et al. 2019), a retrained version of BERT with enhanced training methodology and fine-tune it on our dataset. We use the RoBERTa with these details: RoBERTa-base, 12-layer, 768-hidden, 12-heads, 125M parameters along with Spacy English Transformer NER pipeline[7]. By including the transfer learning [68] paradigm in the standard NER pipeline, our bias recognition model can now identify many other biased words that are not in our dataset. The final output from the bias recognition module is a set of news articles, where the biased words have been identified.

3.4.3 De-biasing module

The de-biasing module is the most important part of Dbias. The purpose of this entire workflow is to reduce the bias effects of news text while maintaining its semantic content, which has a wide range of applications in many domains also. For example, we could convert the biased news headline “Don’t buy the *pseudo-scientific hype* about tornadoes and climate change” to a non-biased sentence as “Don’t buy the

[7] en_core_web_trf

information about tornadoes and climate change". Our third module, de-biasing module is specifically designed to accomplish this goal.

The input to the de-biasing module is a collection of news articles from the bias recognition module with identified biased words, and the output is a list of recommendations for each news article with non-biased or at least less biased words.

Our de-biasing approach comprises of two stages: Bias Masking and Fairness Infill stages. In the bias masking stage, we mask the position of each biased word (token) within each news article. In the fairness infill stage, we fill the masked token positions with new words according to its context in the sentence.

Bias Masking: Our approach to masking is different from the Masked Language Modeling (MLM) [69] task of the standard BERT model. Typically, the MLM task takes a sentence, and randomly masks 15% of the words in the input and then run the entire masked sentence through the model to predict the masked words. In our work, we only mask words that have been flagged as biased by the previous bias recognition module. We demonstrate our approach to bias masking through an example from the news headline [70] in Figure 5.

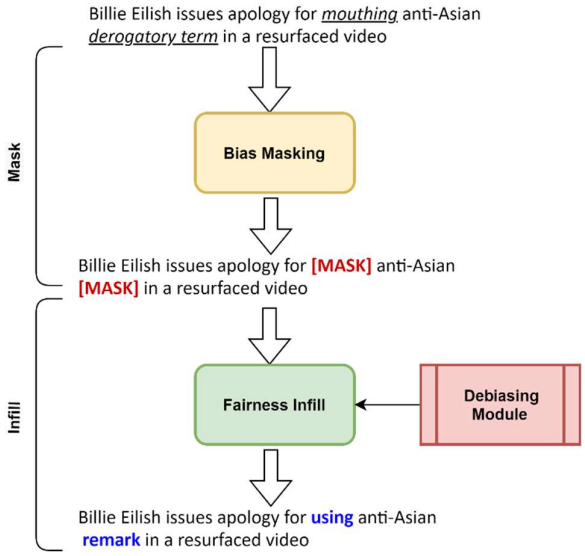

Figure 5: De-biasing example with MLM

As can be seen in Figure 5, we have provided two masks in the news headline. Both of these words have been identified as consisting of biases by the bias recognition module. This approach is different from a typical MLM masking task [71] where only one token is masked and infilled at a time.

Fairness Infilling: We use the MLM technique from the BERT model, however, our masking technique is different from the MLM task of a standard BERT model. We propose a unique mask shifting technique that can mask and unmask more than one token at a time in a sentence. For example, our mask shifting technique can breakdown the above news headline (Figure 5) to two instances (based on two token masks) as below:

- "Billie Eilish issues apology for [MASK] an anti-Asian derogatory term in a resurfaced video."
- "Billie Eilish issues apology for mouthing an anti-Asian [MASK] in a resurfaced video."

Then, both instances will be processed sequentially via mask shifting technique, and each masked token will be filled one at a time before the final sentence is constructed.

Our proposed Fairness Infilling stage can be considered as generalizing the cloze task (Wu et al. 2019) from single tokens to spans of unknown length. Our assumption behind this unique fairness filling is that the new words that are filled in are less or non-biased, which has been validated through our demonstration and experiments.

We recommend a couple of substitute tokens that can be used to infill for each masked token during the Fairness Infilling stage. For example, we can recommend top-k (k=5,10 or so) substitutes for each masked token in the de-biased sentence. We send the top-k recommended sentences (infilled with new tokens) again to the bias detection model (first module) to see the probability of biasness. If the probability of biasness is less than 0.5 or less than the probability of the previous de-biased sentence, we output the sentence as the final output.

4 Experimental setup

Fairness is a complex concept with no one-size-fits-all solution. Considering various definitions [43, 72] and solutions of fairness [6, 73, 74], it is not possible to find one benchmark solution for mitigating biases and to have a fair solution. However, it is still important to evaluate the working of our Dbias pipeline. We evaluate our architecture with two goals in mind: (1) to demonstrate capabilities of our package in terms of bias detection and mitigation algorithms, and (2) to demonstrate how a user can understand the behavior of bias mitigation algorithms on the dataset and make an appropriate choice based on the business need.

4.1 Dataset

In this work, we primarily use MBIC – A Media Bias Annotation Dataset [38]. MBIC is made up of around

1,700 sentences from approximately 1,000 news articles from different news sources (HuffPost, MSNBC, AlterNet, Fox News, Breitbart, USA Today and Reuters and other). The data collection focuses on news articles depicting racial, gender, religious, political, age and related biases, some of these biases are also mentioned in Table 1. The current dataset already significantly extends the existing MBIC data unique and more reliable insights into the perception of bias. In this work, we use the extended version[8] of MBIC dataset with 17,000 records, with 10,000 biased sentences and 7,000 non-biased sentences.

We chose this dataset as our primary source of data because the articles in this dataset cover a broad range of topics (political, scientific, ethnicity and so), all of which are relevant to our objective of identifying various textual biases.

In the original dataset, these biases are identified through crowdsourcing, where the crowdsourced annotators picked the bias-bearing words from the news text. We calculated the biases (gender, race, ethnicity, educational, religion, and language) from the sentences, which are different from the annotators' gender, age, and education labels. Some of the bias-bearing words are taken from this source [75, 76], while others bias-bearing words are added manually: some of these words added by annotators of the MBIC dataset, while we also went through all the record one by one and added more bias-bearing words in the words list. The goal is to have a populated list of biased words from the text that will be later used to train the model.

The dataset features used in this work are:

- *Sentence:* Sentence from the news article
- *News Link:* URL of the news.
- *News Outlet:* News sources (USA today, MSNBC)
- *Topic*: Topic of news (gun-control, coronavirus, white-nationalism and so)
- *Age:* Age of the person annotating the sentence as biased or non-biased
- *Gender*: Gender of annotator
- *Education:* Education of the annotator
- *Biased words:* some of the biased words added by annotators and some are others by us from [75, 76]
- *Label:* News as biased or non-biased

The *age* feature in the dataset is found as the numeric values, however, we categorize it into 3 groups (elder, young, adult). We also categorize the *education* groups based on degrees (high school, undergraduate, graduate) information.

In this work, we consider the following protected attributes from the dataset: '*gender*': [Male, Female], 'age' ['Elder', 'Young', 'Adult'], *'education'*: ['College degree', 'high school'], '*language'*: ['English speaker', 'non-English speaker']. We also consider the following privileged attributes: 'Male' (gender), 'College degree' (education), 'English Speaker' (language). In addition, we consider the following unprivileged attributes: 'Female' (gender), 'high school (education), Non-English Speaker' (language).

Some exploratory analysis on data is shown below:

Figure 6: Biased words in the news articles

Figure 6 shows a word cloud made up of text from the news articles. Among the notable words are "unholy," "radical", "racist", "slammed", "right-wing," and "rage-tweeting," some of which represent political biases during the 2020 US elections.

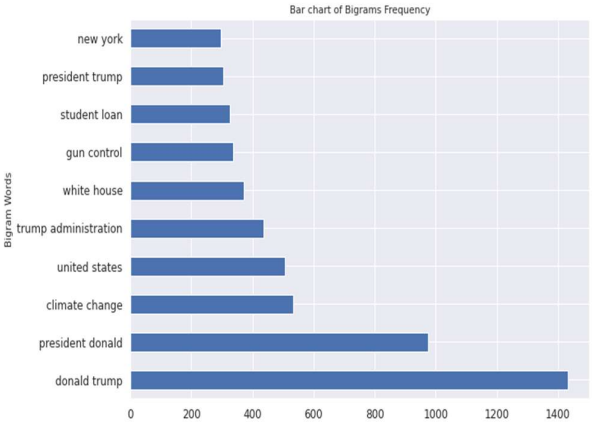

Figure 7 (a): Bigrams of biased words in the news

[8] mbic-a-media-bias-annotation-dataset

Table 2: Baseline methods

Model	Description
Fairness Pre-processing models	
Disparate impact remover [31]	Disparate Impact Remover is a pre-processing approach for increasing fairness between groups (privileged and unprivileged). This technique edits the feature values (e.g., the features that are privileged, unprivileged) so that the data can be made unbiased while preserving relevant information in the data. Once this algorithm has been implemented, any machine learning or deep learning model can be built using the repaired data. The Disparate Impact metric is then used to validate if the model is unbiased (or within an acceptable threshold). In this baseline method, we use a couple of methods using AutoML and reporting the results with the best performing model. For this baseline, the Logistic Regression [77] gave us the best results.
Reweighing [10]	Reweighing is a preprocessing technique that weighs the examples in each group (such as privileged, unprivileged groups) to ensure fairness before classification. This algorithm transforms the dataset to have more equity in positive outcomes on the protected attribute(s) for both the privileged and unprivileged groups. We run a couple of algorithms on the transformed data and report the result with the best performing model, which is Support Vector Machine (SVM) [78] in this experiment.
Fairness In-processing Models	
Adversarial Debiasing [4]	Adversarial debiasing is based on the generative adversarial network (GAN) model proposed by Goodfellow et al. [79]. Through training, this model debiases the word and general feature embeddings. This is an in-processing technique that learns the definitions of fairness, such as demographic parity, equality of odds, and quality of opportunity [72], so that a discriminator (part of GAN) has been tasked with predicting the protected attribute encoded in the bias of the original feature vector, while a competing generator (part of GAN) has been tasked with producing more debiased embeddings to compete with the discriminator.
Exponentiated Gradient Reduction [51]	Exponentiated gradient reduction is an in-processing technique that reduces the fair classification down into a series of cost-sensitive classification problem. It also returns a randomized classifier with the lowest empirical error (approximation of the expected error), if the fair classification rules are met.
Fairness post-processing models	
Calibrated Equalized Odds Postprocessing [23]	Calibrated equalized odds postprocessing is a postprocessing technique that optimizes over calibrated classifier score outputs to find probabilities for changing output labels with an equalized odds objective.
Equalized Odds postprocessing [24]	Equalized odds postprocessing is a post-processing technique that uses a linear program to find probabilities for changing output labels in order to optimize equalized odds.
Our approach	
Dbias	Our approach mitigates biases during the pre-processing stage and ensures that fairness is carried on throughout the ML pipeline to give a fair representation of data.

Figure 7 (a) and (b) shows the top biased bigrams and trigrams respectively from the news articles.

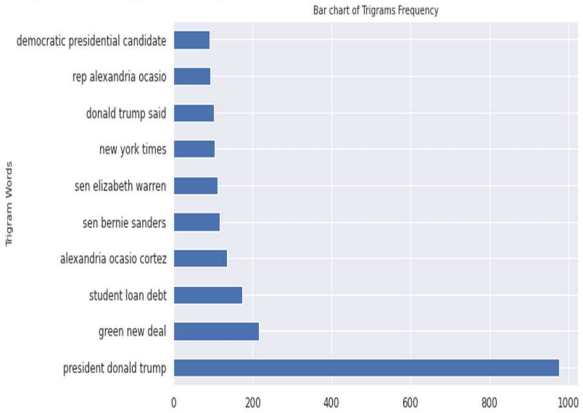

Figure 7 (b): Trigrams of biased words in the news

Some of the biased words that we see in these news articles are related to Elections 2020, COVID-19, climate change, students' loan and so on.

4.2 Baselines Methods

We could not find a single state-of-the-art model that can perform all these tasks: (1) bias detection; (2) bias recognition (3) bias masking, mitigation and recommendation, altogether. Thus, we use the fairness baseline methods as categorized in the related work [17]. These categories are based on the stages of the ML pipeline at which bias is detected and mitigated, which are briefly defined as:

- *Fairness pre-processing* methods are concerned with the early stages of bias detection and mitigation. It preprocesses and converts the data

into non-biased inputs and make it ready for the machine learning training.
- *Fairness in-processing* methods focus on fairness interventions during the model training process.
- *Fairness post-processing*: these methods focus on fairness interventions after the data has been pre-processed and the model has been trained.

In addition to comparison with baselines (Table 2), we also use other baseline methods to evaluate the effectiveness of individual modules of Dbias, which are detailed in their respective sections.

4.3 Evaluation strategy

Our evaluation strategy is given below:

***Evaluating the Dbias against state-of-the-art baselines*:** To our knowledge, there is no single model or pipeline that can perform all three tasks simultaneously: bias detection, bias recognition, and de-biasing. By examining and implementing these baseline methods (Table 2), we discover that most of them are based on a task for bias detection and mitigation. Thus, we evaluate these baselines and our Dbias pipeline, with a particular emphasis on the task of bias detection and mitigation. This evaluation strategy is inspired by the AIF 360 [6] evaluation technique. The outline of this evaluation is:

- First, we evaluate the ability of each method in detecting the biases from the data using the accuracy measure.
- Second, we de-bias the data using each of the method's specific approach. This de-biased data is a transformed data from the original data.
- Third, we evaluate the ability of each method on fairness (i.e., how many biases are mitigated by each method). We also test each method again to detect the remaining biases in transformed data.

Evaluating the effectiveness bias detection module of Dbias: We evaluate the performance of Dbias for the bias detection module. We evaluate various classification, Transformer-based and embedding models with our fine-tuned DistilBERT to determine which model/setting yields the most accurate results.

***Evaluating the effectiveness bias recognition module of Dbias*:** We evaluate the performance of Dbias for the bias recognition module. We compared several configurations of different NER models to test which setting gives us the most best results.

***Evaluating the effectiveness masking technique of Dbias*:** We explore the influence of different masking probability to find the best setting for the masking.

The purpose of all this evaluation strategy is to compare Dbias pipeline and its individual modules to state-of-the-art methods.

4.4 Evaluation metrics

To assess the performance of our proposed model, we use the accuracy (ACC), precision (PREC), recall (Rec), F1-score (F1), and Disparate Impact (DI) as the evaluation metrics. A confusion matrix determines the information about actual and predicted values, as shown in Table 3.

Table 3: Confusion Matrix

	Actual Fake	**Actual Real**
Predicted Fake	TP	FP
Predicted Real	FN	TN

The variables TP, FP, TN, and FN in the confusion matrix refer to the following:

- *True Positive (TP):* number of biased news that are identified as biased.
- *False Positive (FP):* number of unbiased news that are identified as fake news.
- *True negative (TN):* number of unbiased news that are identified as unbiased news.
- *False negative (FN):* number of biased news that are identified as unbiased news.

For the Prec, Rec, F1 and ACC, we perform the specific calculation as shown in the Equation (1), (2), (3) and (4) respectively:

$$Prec = \frac{TP}{TP + FP} \quad (1)$$

$$Rec = \frac{TP}{TP + FN} \quad (2)$$

$$F1 = \frac{TP}{TP + \frac{1}{2}(FN + FP)} \quad (3)$$

$$ACC = \frac{TP + TN}{TP + TN + FP + FN} \quad (4)$$

Disparate Impact (DI) [33] is an evaluation metric to evaluate fairness. It compares the proportion of individuals that receive a positive output for two groups: an unprivileged group and a privileged group. The industry standard for DI is a four-fifths rule [80] which means if the unprivileged group receives a positive outcome less than 80% of their proportion of the privileged group, this is a disparate impact violation. An acceptable threshold should be between 0.8 and 1.25, with 0.8 favoring the privileged group, and 1.25 favoring the unprivileged group [81]. Mathematically, it can be defined as:

Table 3: Comparison of our framework with the baseline methods.

		Before de-biasing					*After debiasing*				
	Model	**PREC**	**REC**	**F1**	**ACC**	**DI**	**PREC**	**REC**	**F1**	**ACC**	**DI**
Pre	Disparate impact remover	0.593	0.549	0.570	0.587	0.702	0.532	0.414	0.466	0.541	0.804
	Reweighing	0.613	0.535	0.572	0.619	0.702	0.591	0.524	0.555	0.604	0.832
In	Adversarial Debiasing	0.624	0.600	0.612	0.641	0.702	0.592	0.587	0.590	0.610	0.923
	Exponentiated Gradient Reduct.	0.612	0.587	0.599	0.626	0.702	0.589	0.557	0.573	0.606	0.896
Post	Calibrated Equalized Odds	0.568	0.479	0.520	0.560	0.702	0.563	0.479	0.518	0.523	0.829
	Equalized Odds	0.498	0.487	0.492	0.577	0.702	0.487	0.488	0.487	0.505	0.818
	Dbias	**0.735**	**0.784**	**0.759**	**0.776**	0.702	**0.690**	**0.704**	**0.697**	**0.743**	**1.012**

$$Disparate\ impact\ ratio = \frac{(num\_positives(privileged=Fals\ ) / num\_instances(privileged=\ \ \ \ ))}{(num\_positives(privileged=True) / num\_instances(privileged=True))} \quad (5)$$

where num_positives are the number of individuals in the group: either privileged=False (unprivileged), or privileged=True (privileged), who received a positive outcome. The num_instances are the total number of individuals in the group.

The DI calculation is the proportion of the unprivileged group that received the positive outcome divided by the proportion of the privileged group that received the positive outcome. This means DI ratio is the ratio of positive outcomes (Bias=1) in the unprivileged group (in our case, females, elderly, non-English speakers, no higher education) divided by the ratio of positive outcomes in the privileged group (males, adults, English speakers, higher education).

4.5 Common Hyperparameter

The common hyperparameters that we use in Dbias are a batch size of 16, 10 epochs, sequence length 512, number of labels for news is 2 (bias, non-biased). The learning rate, warm-up setups, the drop-out rate and other parameters for each module are optimized according to their best settings. For fair comparison, we tune all the other methods (baselines) to their optimal hyperparameter settings and report the best results.

5 Results and Analysis

The results and analysis are shown and discussed here.

5.1 Comparison between baselines and Dbias

These experiments are conducted in two-fold manner: (1) evaluation before de-biasing, and (2) evaluation after the debiasing.

In the evaluation before de-biasing, we first calculate the DI metric that simply takes the values of protected variables to measure the DI ratio. Then we run the classification methods provided by each method to test the ability of each method in detecting the biases from the data. Some of these baseline methods do not have a classification method (the pre-processing ones), so, we use AutoML[9] to select the best performing classification method. Finally, we evaluate the performance of each method for the classification accuracy.

In the evaluation after de-biasing, we apply the bias mitigation method provided by each method on the original data to create a transformed dataset. The transformed data is assumed to be a fairer dataset because the transformation is learned as a new representation of the data using some mapping or project functions. Lastly, we compute all the metrics (PREC, REC, F1, ACC and DI) on the transformed dataset.

Next, we present the results of all methods (baseline and Dbias) in Table 3.

Overall results: Overall, the results in Table 3 shows that our approach outperforms all the baseline methods in detecting the biases. This is demonstrated by our model having the highest PREC, REC, F1, and ACC scores when compared to the baseline models in the 'before de-biasing' testing. The DI ratio of all models in the 'before de-biasing' evaluation phase is constant. This is because the DI is calculated based on the original dataset before we apply any technique to our dataset.

In the 'after de-biasing' evaluation, our approach also performs its best to identify the biases. In this phase, our model's DI ratio is also a good value. A good DI value is one that is between 0.8 and 1.25, ensuring that different groups (gender, race, education, and such) are balanced [15, 33, 81]. Our model has a DI ratio of 1.012, which indicates we are able to mitigate biases among various groups in an appropriate and balanced manner.

[9] automl

Baselines comparison: Among the baselines, the general performance of the fairness in-processing methods is better than the pre-processing methods, which is better than the post-processing methods. This is most likely due to the fact that the fairness in-processing models have explicit control over the optimization function of a model. As a result, these models can better optimize the measure of fairness during the model training. The pre- and post-processing approaches do not change a model explicitly. This means that, in their current state, any ML library can be used for model training in pre- and post-processing approaches. However, this comes at the cost of having no direct control over the optimization function of the ML model. Thus, each technique has a tradeoff, depending on whether we need more explicit control over the method (as in in-processing) or whether we need to incorporate fairness without affecting model training (as in pre- and post-processing).

Among the baselines, the in-processing methods provide much fairer results by mitigating the biases. In-processing methods work by incorporating one or more fairness measures into the model optimization functions in order to converge on a model parameterization that maximizes both performance and fairness [17]. Between the two in-processing baselines, we see that the performance of Adversarial Debiasing [4] is better than the Exponentiated Gradient Reduction method [51]. This is demonstrated by the higher precision, recall, F1 and accuracy scores of Adversarial Debiasing method when compared to the other method in both phases of the evaluation (before and after debiasing). Adversarial Debiasing has a DI ratio of 0.923, while Exponentiated Gradient Reduction has a ratio of 0.896. A value close to 0.8 indicates that the privileged group benefits more.

Based on above results, the Adversarial Debiasing produces more fair predictions (a value between 0.8 and 1.25 i.e., close to 1) between the two in-processing techniques. The Adversarial Debiasing methods uses the adversarial training method to enforce a fairness constraint during the model optimization. The Exponentiated Gradient Reduction yields a randomized classifier with the lowest (empirical) error subject to the desired fairness constraints. This result indicates that adversarial training can be a useful technique to ensure fairness in NLP solutions.

Next, comes the performance of pre-processing methods. Pre-processing methods usually change the sample distributions of protected variables or perform transformations on the data to make it less discriminatory in the training set [15, 17]. In this case, the main idea is to train a model on a 'repaired' dataset. Between the two pre-processing methods among our baselines, we see that the general performance of Reweighing [10] is better than the Disparate impact remover method [31]. This is shown with the better scores of Reweighing during both evaluation phases. The DI ratio of Reweighing after de-biasing is 0.832, which is better than that of the Disparate impact remover method and also better than its own 'before debiasing' evaluation score. However, a value of 0.832 is still close to 0.8, which means that it is favoring privileged groups.

Last comes the performance of post-processing methods. Post-processing methods tend to apply transformations to model predictions to improve fairness [17]. Between the two post-processing methods, the general performance of Calibrated Equalized Odds [23] is better than the Equalized Odds method [24]. This is shown with the better scores of Calibrated Equalized Odds during both evaluation phases. The DI ratio of Calibrated Equalized Odds is 0.829, which is a better value than the other method but still not close to 1 (a value around 1 means balancing between different groups).

Compared to the baselines, our method is able to achieve the highest classification accuracy during both evaluation modes. The DI ratio of our approach is also close to 1, which shows that we are able to achieve a balance between unprivileged and privileged groups.

Tradeoff between accuracy and fairness*:* Overall, these results indicate that there is a tradeoff between accuracy and the fairness (DI ratio) measures. Usually, when one goes high, the other metric value goes down, as demonstrated in the previous research also [6, 17, 43]. However, a good model is one that is able to achieve a good value of DI ratio without much loss in accuracy. Our Dbias model achieves highest accuracy scores in 'before-de-biasing' and it does not lose much accuracy in 'after de-biasing' testing. It also achieves a good balance between these two measures. The superiority of our approach is attributed to its fair ML design that is able to detect, recognize and mitigate bias in a workflow. In addition to having a good accuracy and fairness values, our model can also mitigate with many other kinds of biases from the data, which are not covered by the other models.

5.2 Effectiveness of the Bias Detection Module

In this experiment, we evaluate the performance of our framework for the bias detection module (the first module of the Dbias). We fine-tune the bias detection module using different models and embeddings. The goal is to see which model gives us the best results for the classification task. We also compare these methods with our fine-tuned DistilBERT model.

We use the following models in this experiment. Some of these methods are traditional ML methods and some are deep neural network methods, we also use the Transformer-based methods in this experiment, which are advanced deep neural methods with self-attention [82].

Logistic Regression-TFIDF Vectorization (LG - TFIDF): We use the Logistic Regression (LG) [77] with TfidfVectorizer[10] word embedding method [83]. TFIDF Vectorization converts a collection of raw documents to a matrix of TF-IDF features. Logistic regression + TFIDF Vectorization has shown to be a good baseline method for many classifications tasks, such as hate speech detection [84], text classification [85]. This is a classical ML approach.

Random Forest + TFIDF Vectorization (RF-TFIDF): We use the Random Forest (RF) classifier [86] with TfidfVectorizer word embedding method. Random Forest is a classifier that uses a number of decision trees on different subsets of a dataset to improve predictive accuracy. Random Forest + TF-IDF Vectorization are also used for text classification, sentiment analysis and related tasks [86, 87], so it can serve as good method for bias detection. This is a classical ML approach.

Gradient Boosting Machine + TFIDF Vectorization (GBM-TFIDF): We use the Gradient Boosting Machine (GBM) [88] with TfidfVectorizer word embedding method. GBM is forward learning ensemble method that has shown good predictive results through increasingly refined approximations [88]. This is a classical ML approach.

Logistic Regression + ELMO (LG-ELMO): We use LG with ELMO embeddings. ELMo is a contextual word embedding technique based on bi-directional LSTM [89]. Unlike TFIDF methods, which look at a dictionary of words and build corresponding vectors, ELMo looks at the entire sentence before assigning each word in the embedding. However, compared to TFIDF, creating ELMO embeddings is a computationally heavy task. LG and ELMO can be used for a variety of classification tasks. This is a mix of classical ML approach (LG method) with deep neural embeddings.

MultiLayer Perceptron + ELMO (MLP- ELMO): We also use the MLP[11], a feedforward artificial neural network with ELMO embeddings, which has shown good performance for the classification tasks [90, 91]. This is a deep neural network-based method.

Bert-base: Bidirectional Encoder Representations from Transformers (BERT) is a deep neural network model pretrained on a large corpus of English data in a self-supervised fashion [55], which has shown state-of-the-art performance in a variety of classification tasks [92]. We use the bert-based uncased[12] in this work. This is a Transformer-based approach.

***RoBERTa-base*:** Robustly Optimized BERT Pre-training Approach (RoBERTa) [93] optimizes the training of BERT with improved training methodology. RoBERTa has also shown good performance in many classification tasks, including text classification [94]. This is a Transformer-based approach.

***DistillBERT*:** We also use the DistilBERT [95], which is a small, fast, cheap and light Transformer model trained by distilling BERT base. It has 40% less parameters than bert-base-uncased, runs 60% faster while preserving over 95% of BERT's performances. This is a Transformer-based approach.

The results for the evaluation of bias detection module are shown in Table 4.

Table 4: Effectiveness of different classification models for bias detection task

Model	PREC	REC	F1
LG-TFIDF	0.62	0.61	0.61
RF-TFIDF	0.65	0.64	0.64
GBM - TFIDF	0.65	0.66	0.65
LG- ELMO	0.66	0.68	0.67
MLP- ELMO	0.69	0.67	0.68
Bert-base	0.72	0.69	0.70
RoBERTa-base	0.75	0.70	0.72
Distilbert	**0.76**	**0.74**	**0.75**

Overall, the results in Table 4 show the better performance of deep neural network embeddings (i.e., ELMo) compared to TFIDF vectorization when used with LG, RF and GBM. The deep neural embeddings and the deep neural methods (MLP, BERT, RoBERTa and DistilBERT) also perform better than the traditional ML methods.

[10] TfidfVectorizer
[11] MLP
[12] bert-base-uncased

There is a slight performance difference among the three classical ML approaches (LG-TFIDF, RF-TFIDF, and GBM-TFIDF), with GBM-TFIDF marginally performs better than the other two models. This is demonstrated by GBM- TFIDF's performance, which results in a 1% higher in F1-score when compared to RF-TFIDF and a 4% increase when compared to LG-TFIDF.

Using ELMO embeddings, we can see that the LG model's performance has improved by about 5% compared to LG-TFIDF. This is demonstrated by LG-ELMO having a higher F1-score (0.70) than LG-TFIDF (0.65). Additionally, the LG-ELMO outperforms other traditional machine learning approaches by approximately 1-6 % better F1- scores.

The results in Table 4 also show that the deep neural baseline MLP-ELMO performs better than traditional ML baselines as well as the LG-ELMO (a mix of classical ML and deep neural network embedding model). The result shows that deep neural embeddings, in general, can outperform traditional embedding method (e.g., TFIDF) in text classification tasks (e.g., the bias detection in this work). This is probably because, deep neural embeddings can better capture the context of the words in the text. The use of deep neural embeddings in conjunction with a model based on deep learning, such as MLP, further improves the results, as demonstrated by the better performance of MLP-ELMO model compared to ML baselines and LG-ELMO. Though the deep neural network methods perform better in many NLP tasks, the traditional ML methods are usually faster and computationally less expensive, which is also discussed in the literature [96].

We also use the Transformer-based embeddings, such as from BERT, which are on dynamic word embeddings. Transformers are large encoder-decoder models that employ a sophisticated attention mechanism to process an entire sequence [82]. The results show that Transformer-based methods outperform the other methods (ML and simple deep learning-based methods) in the bias detection task. Among the Transformer-based approaches, RoBERTa outperforms the BERT model by approximately 2% in terms of F1-score, while DistilBERT outperforms the RobBERTa model by approximately 3%.

DistilBERT is a smaller, faster, and lighter than BERT and RoBERTa. When we apply DistilBERT to our dataset, it also performs significantly better than all the other models, as shown in Table 4. As a result, we choose to work with the DistilBERT in our bias detection module,

5.3 Effectiveness of bias recognition module

We also test the effectiveness of our biased recognition module. We compared several configurations of NER models, such as ML-based NER pipelines, Transformer-based approach until we obtained a configuration that we consider to be the best for our goals. Next, we see the performance of different NER pipelines for the bias recognition task:

Spacy core web small (sm) pipeline*[13] *(core-sm): This pipeline consists of following components: token-to-vector, tagger part-of-speech tagging, parser, Sentence Recognizer, Named Entity Recognition, attribute ruler and a lemmatizer. It is an English pipeline trained on written web text (blogs, news, comments), which includes vocabulary, syntax and entities. It is called as 'core' as it is based on traditional ML and packaged in the Spacy Core library, which use a CPU-optimized pipeline. The 'small' in this pipeline refers to the size of mode, which is 13 MB in this case.

Spacy core web medium (md) pipeline*[14] *(core-md): This is the same pipeline as core-sm but with medium size, which is 43 MB. The model size refers to different configurations, e.g., more data points, different parameters, iterations, vector size and such.

***Spacy core web large (lg) pipeline*[15] *(core-lg*):** This is the Spacy core NER pipeline like as core-sm and as core-md but with large size, which is 741 MB.

Spacy core web transformer (trf) pipeline*[16]*(core-trf): This is spacy core NER pipeline, but it has Transformer as the vectorizer. The difference between core-trf and other core pipelines is in the embedding model. The core-trf uses the RoBERTa [93], Transformer as embedding model that helps to automatically identify and extract entities from the text. The results of different NER pipelines to recognize the biases are shown in Table 5.

Table 5: Evaluation of different NER pipelines for the bias recognition task

Model	PREC	REC	F1	ACC
Core-sm	0.59	0.27	0.37	0.37
Core-md	0.61	0.45	0.52	0.53
Core-lg	0.60	0.62	0.60	0.67
Core-trf	**0.66**	**0.65**	**0.63**	**0.72**

[13] en_core_web_sm
[14] en_core_web_md
[15] en_core_web_lg
[16] en_core_web_trf

The results in Table 5 show that core-trf outperforms all the other NER methods in terms of precision recall, F1-score and accuracy. This is because a Transformer-based model can identify entities and relations within the text and can generate a text representation that takes the context of each term into account. Additionally, these findings indicate that model performance in terms of accuracy metrics improves as model size increases. This is likely due to the fact that the larger model contains a greater number of parameter settings and data points, all of which affect the model's predictive performance. However, these benefits come at the expense of resource utilization, memory, CPU cycles, and latency delay. Based on these results, we choose to work with core-trf to recognize the bias-bearing words from the news articles.

5.4 Effectiveness of masking technique

In this section, we explore the influence of masking in the MLM technique. We use different settings for the mask: we replace 5% of the input sequence with [MASK] with a probability *p* of 0.1, 0.3, 0.5, 0.8 and 1.0, as in related works [55, 69]. There is no standard masking percentage, but we use 5% based on the sentence length (around 10-40 words) of news text in our dataset. We then compare these settings with our masking technique where we only replace the bias-bearing words and we don't make use of any random probability as in typical MLM tasks [55, 69]. The goal of this experiment is to see if using different masking techniques (with different probabilities) affects the final output. The results of our model with different MLM settings are shown in Figure 8.

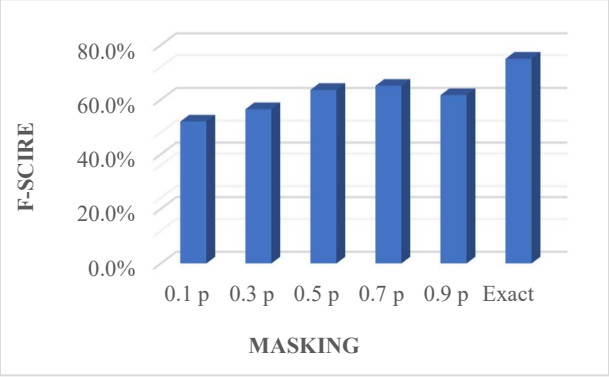

Figure 8: probability *p* vs exact bias mask

The results in Figure 8 show that the performance of Dbias improves when probability *p* increases from 0.1 till 0.7. When *p* is too small, the model perhaps tends to overfit, since our model has many parameters. Thus, the performance is not optimal. However, when *p* is too large i.e., 0.9, the performance of model starts to decline. In contrast, our exact masking of biased words shows the best performance. The general conclusion that we can draw from these results is that if we replace the input sequence with [MASK] using varying probabilities, the model performance may be affected, as the model may be learning to detect only the masked word, rather than actual word representations. As our use case in this work is masking the biased words only from the text, so an exact match can give us better results.

5.5 Working of Dbias model

We release our Dbias package [17] that is used to detect and mitigate biases in NLP tasks. The model tasks are summarized as shown in Table 6:

Table 6: Dbias tasks

Feature	Output
Text Debiasing	Returns debiased news recommendations with bias probability
Bias Classification	Classifies whether a news article is biased or not with probability
Bias Words/Phrases Recognition	Extract Biased words or phrases from the news fragment
Bias masking	Returns the news fragment with biased words masked out

The model can be installed using the commands:

- `pip install Dbias`
- `pip install https://huggingface.co/d4data/en_pipeline/resolve/main/en_pipeline-any-py3-none-any.whl`

The input to the model can be any sentences that may contain biased words and we get the de-biased output. We show the working our model in Figure 9.

As illustrated in Figure 9, given a news article or any text that may contain biased words, our model can determine whether or not the text is biased. This is made possible by the first module: bias detection module. The output is then forwarded to the next module, namely the bias recognition module, which identifies bias-bearing words. The text with identified biased words is then sent to the de-biasing module, which masks the biased words and makes suggestions for new words to replace them. The final output is a set of non-biased or at least minimally biased sentences for each input sentence.

[17] https://pypi.org/project/Dbias/

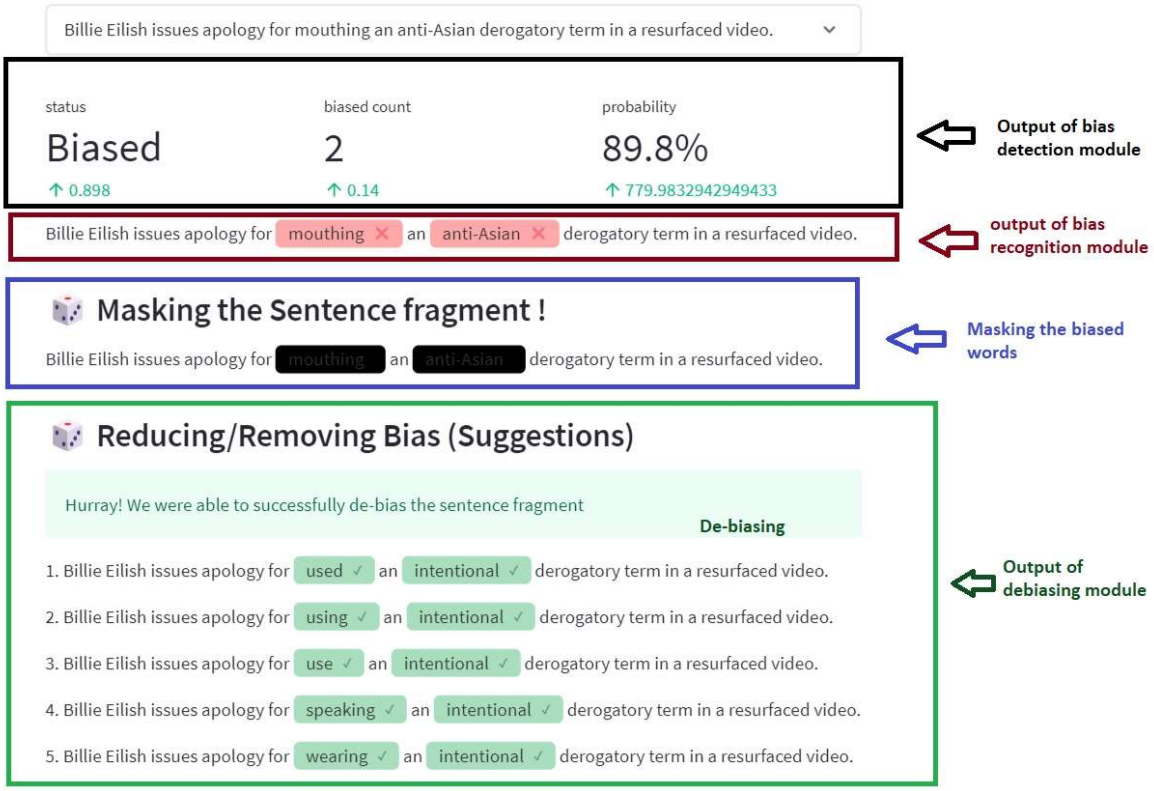

Figure 9: Working of Dbias

We also show the bias probability that is achieved from an original news article to a de-biased article in Figure 10. We perform a forward pass to compute logits and apply softmax to calculate probabilities. A lower score, here, means that less bias exists and therefore detected less.

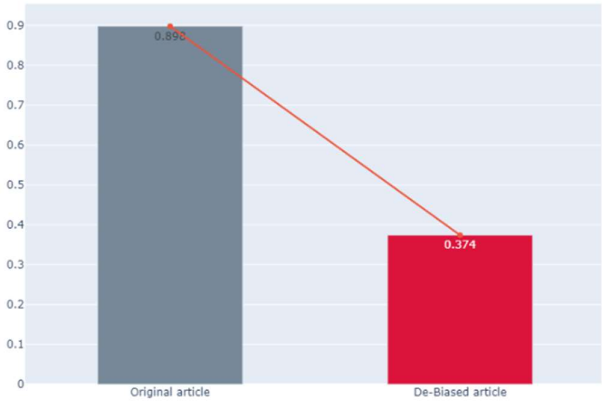

Figure 10: Bias probability in a news article

As shown in Figure 10, our Dbias framework can reduce the biasness of a news article to about 50%.

6 Discussion

Fairness in ML and AI is a relatively new but rapidly growing field of study in fields such as information retrieval, recommender systems, and so on. The broader literature on fairness in ML is a crucial starting point, therefore, we undertook this research with great care and compassion to avoid any pitfalls that result in overbroad or ungeneralizable claims.

As with any research study, there are limitations. In order to help our readers understand and generalize the results, we have taken extra care to disclose the limitations of the data, metrics, and methods used. There are some limitations that we want to discuss here, and we also discuss some future recommendations.

Different definitions of biases and fairness: There is currently no universally accepted definition of what constitutes bias and fairness [14, 15]. The findings drawn about one bias cannot be generalized to other biases. Additionally, biases manifest in a variety of ways (e.g., the bias definition of gender cannot be applied to ethnicity or social status). While more standard definitions of biases and fairness may be discovered in the future, we must first investigate a wide variety of biases in different applications to determine the fairness of data and algorithms.

Biases evolve over time: While much of the research in this field considers a small number of biases (social, demographic, and so on), we emphasize that there are

many other types of biases that are overlooked in the research. These biases also evolve over time [41]. For instance, numerous biases have arisen in recent years as a result of COVID-19 or U.S ELECTIONS 2016 and 2020. Taking this limitation into account, we develop a system capable of dynamically removing biases from data. This dynamic definition [14] states that an initially fair system may become unfair over time if users respond in a biased manner, or that it may evolve toward a fairer system if users respond positively to recommendations that improve overall fairness. The fundamental requirement for our system is to fine-tune it using the domain data. While this is a start, we may need to investigate additional domains to enrich the Dbias pipeline.

Fairness metrics*:* There are many fairness metrics, for example, statistical parity [97], predictive parity [98], calibration [45], pairwise fairness [99] and other [100], in the literature. There is still much work left to do to understand how best to apply and interpret these fairness metrics in our study.

Use of appropriate data*:* Data collection is a significant challenge for fairness research because it frequently requires sensitive data that cannot be collected via standard information retrieval or recommender system data sets. We use a manually annotated news dataset in this study to identify bias-bearing words. We encourage researchers to use unstructured data to identify and mitigate more biases in the text without focusing exclusively on specific data points (e,.g., race, gender, education).

Transfer learning: In our Dbias modules, we used the transfer learning technique. We recognize that transfer learning may introduce additional biases. However, we find that carefully fine-tuning the models on the appropriate data can be useful. For example, we fine-tune our modules on news data and then these models help us to detect and mitigate various biases from the news data.

Scarcity of labelled biased data: One of the research's limitations is the scarcity of labelled biased data. So far, we have relied on the MBIC dataset to assist us in detecting and mitigating bias, which has been validated through extensive experiments. However, we would like to acquire more labelled data in the future to train our package.

There is still much work to be done toward achieving fairness in ML, and we hope that others in the research community will continue to contribute their own approaches to fairness and bias checking, mitigation, and explanation to the toolkit. This package is developed to de-bias news articles, anyone can use it to train on other types of data, such as journalism, hiring applications, prison sentencing or health science, and then use it to de-bias that data. As a result, one future direction is to extend the toolkit's application to additional datasets and scenarios.

7 Conclusion

We build the Dbias, a pipeline for fair ML, which is composed of three main modules: bias detection, bias recognition, and de-biasing. We develop a Transformer-based model to detect biased news using labelled news data; we develop a named entity recognition model based on the Transformer architecture to identify biased words in biased news articles; and we use the masked language modelling technique to replace the biased words in the text with neutral words. We compare Dbias' performance to state-of-the-art fairness methods. Additionally, we evaluate the efficacy of individual components of Dbias. We release Dbias as a freely downloadable package for the users and practitioners. This research serves as a forum for researchers interested in de-biasing the text. This package can be used by developers to detect and mitigate bias.

Compliance with Ethical Standards

Conflict of Interest: On behalf of all authors, the corresponding author states that there is no conflict of interest.

References

1. Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., Kuksa, P.: Natural language processing (almost) from scratch. J. Mach. Learn. Res. 12, 2493–2537 (2011)
2. Azure, M.: Artificial Intelligence vs. Machine Learning | Microsoft Azure, https://azure.microsoft.com/en-us/overview/artificial-intelligence-ai-vs-machine-learning/#introduction
3. Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I.: Language models are unsupervised multitask learners. OpenAI blog. 1, 9 (2019)
4. Zhang, B.H., Lemoine, B., Mitchell, M.: Mitigating unwanted biases with adversarial learning. In: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society. pp. 335–340 (2018)
5. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: A Survey on Bias and Fairness in Machine Learning. ACM Comput. Surv. 54, (2021). https://doi.org/10.1145/3457607
6. Bellamy, R.K.E., Mojsilovic, A., Nagar, S.,

Ramamurthy, K.N., Richards, J., Saha, D., Sattigeri, P., Singh, M., Varshney, K.R., Zhang, Y., Dey, K., Hind, M., Hoffman, S.C., Houde, S., Kannan, K., Lohia, P., Martino, J., Mehta, S.: AI Fairness 360: An extensible toolkit for detecting and mitigating algorithmic bias. IBM J. Res. Dev. 63, (2019). https://doi.org/10.1147/JRD.2019.2942287

7. Jeffrey Dastin: Amazon scraps secret AI recruiting tool that showed bias against women, https://www.reuters.com/article/us-amazon-com-jobs-automation-insight-idUSKCN1MK08G
8. The Guardian: Facebook charged with housing discrimination in targeted ads | Facebook | The Guardian, https://www.theguardian.com/technology/2019/mar/28/facebook-ads-housing-discrimination-charges-us-government-hud
9. Obermeyer, Z., Powers, B., Vogeli, C., Mullainathan, S.: Dissecting racial bias in an algorithm used to manage the health of populations. Science (80-.). 366, 447–453 (2019)
10. Kamiran, F., Calders, T.: Data preprocessing techniques for classification without discrimination. (2012)
11. Raza, S., Ding, C.: News recommender system: a review of recent progress, challenges, and opportunities. Artif. Intell. Rev. 1–52 (2021). https://doi.org/10.1007/s10462-021-10043-x
12. Raza, S., Ding, C.: News Recommender System Considering Temporal Dynamics and News Taxonomy. In: Proceedings - 2019 IEEE International Conference on Big Data, Big Data 2019. pp. 920–929. Institute of Electrical and Electronics Engineers Inc. (2019)
13. Ribeiro, F.N., Henrique, L., Benevenuto, F., Chakraborty, A., Kulshrestha, J., Babaei, M., Gummadi, K.P.: Media bias monitor: Quantifying biases of social media news outlets at large-scale. In: Twelfth international AAAI conference on web and social media (2018)
14. Ekstrand, M.D., Das, A., Burke, R., Diaz, F.: Fairness and Discrimination in Information Access Systems. 1–98 (2021)
15. Caton, S., Haas, C.: Fairness in Machine Learning: A Survey. 1–33 (2020)
16. Dacon, J., Liu, H.: Does Gender Matter in the News? Detecting and Examining Gender Bias in News Articles. 385–392 (2021). https://doi.org/10.1145/3442442.3452325
17. Nielsen, A.: Practical Fairness. O'Reilly Media (2020)
18. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and Removing Disparate Impact *. https://doi.org/10.1145/2766XXX.XXXXXXX
19. Zemel, R., Wu, Y., Swersky, K., Pitassi, T., Dwork, C.: Learning fair representations. In: International conference on machine learning. pp. 325–333 (2013)
20. Wexler, J., Pushkarna, M., Bolukbasi, T., Wattenberg, M., Viégas, F., Wilson, J.: The what-if tool: Interactive probing of machine learning models. IEEE Trans. Vis. Comput. Graph. 26, 56–65 (2019)
21. Donahue, J., Krähenbühl, P., Darrell, T.: Adversarial feature learning. arXiv Prepr. arXiv1605.09782. (2016)
22. Raza, S., Ding, C.: A Regularized Model to Trade-off between Accuracy and Diversity in a News Recommender System. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 551–560 (2020)
23. Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., Weinberger, K.Q.: On fairness and calibration. arXiv Prepr. arXiv1709.02012. (2017)
24. Hardt, M., Price, E., Srebro, N.: Equality of opportunity in supervised learning. Adv. Neural Inf. Process. Syst. 29, 3315–3323 (2016)
25. Dixon, L., Li, J., Sorensen, J., Thain, N., Vasserman, L.: Measuring and mitigating unintended bias in text classification. In: Proceedings of the 2018 AAAI/ACM Conference on AI, Ethics, and Society. pp. 67–73 (2018)
26. Holstein, K., Wortman Vaughan, J., Daumé III, H., Dudik, M., Wallach, H.: Improving fairness in machine learning systems: What do industry practitioners need? In: Proceedings of the 2019 CHI conference on human factors in computing systems. pp. 1–16 (2019)
27. Kirkpatrick, K.: It's not the algorithm, it's the data. Commun. ACM. 60, 21–23 (2017)
28. Xu, F., Shao, G.: Building Scalable Streaming Pipelines for Near Real-Time Features, (2021)
29. WeAreNetflix: Netflix Research: Analytics, https://research.netflix.com/research-area/recommendations%0Ahttps://research.netflix.com/research-area/recommendations%0Ahttps://www.youtube.com/watch?v=sAQmj8a_aI8&feature=emb_logo
30. Labs, J.S.: John Snow Labs | NLP & AI in Healthcare, https://www.johnsnowlabs.com/
31. Feldman, M., Friedler, S.A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: Certifying and removing disparate impact. In: proceedings of the 21th ACM SIGKDD international conference on knowledge discovery and data mining. pp. 259–268 (2015)
32. Mehrabi, N., Gowda, T., Morstatter, F., Peng, N., Galstyan, A.: Man is to person as woman is to location: Measuring gender bias in named entity recognition. Proc. 31st ACM Conf. Hypertext Soc. Media, HT 2020. 231–232 (2020). https://doi.org/10.1145/3372923.3404804
33. Tramèr, F., Atlidakis, V., Geambasu, R., Hsu, D., Hubaux, J.P., Humbert, M., Juels, A., Lin, H.: FairTest: Discovering Unwarranted Associations in Data-Driven Applications. Proc. - 2nd IEEE Eur. Symp. Secur. Privacy, EuroS P 2017. 401–416 (2017). https://doi.org/10.1109/EuroSP.2017.29
34. Adebayo, J.A., others: FairML: ToolBox for diagnosing bias in predictive modeling, (2016)
35. Bantilan, N.: Themis-ml: A fairness-aware machine learning interface for end-to-end discrimination

discovery and mitigation. J. Technol. Hum. Serv. 36, 15–30 (2018)
36. Kamiran, F., Karim, A., Zhang, X.: Decision theory for discrimination-aware classification. In: 2012 IEEE 12th International Conference on Data Mining. pp. 924–929 (2012)
37. Biega, A.J., Gummadi, K.P., Weikum, G.: Equity of attention: Amortizing individual fairness in rankings. In: The 41st international acm sigir conference on research \& development in information retrieval. pp. 405–414 (2018)
38. Spinde, T., Rudnitckaia, L., Sinha, K., Hamborg, F., Gipp, B., Donnay, K.: MBIC – A Media Bias Annotation Dataset Including Annotator Characteristics. Proc. iConference 2021. 1–8 (2021)
39. Baly, R., Karadzhov, G., Alexandrov, D., Glass, J., Nakov, P.: Predicting factuality of reporting and bias of news media sources. In: Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, EMNLP 2018. pp. 3528–3539 (2020)
40. Bruns, A.: It's not the technology, stupid: How the 'Echo Chamber'and 'Filter Bubble'metaphors have failed us. Int. Assoc. Media Commun. Res. (2019)
41. Raza, S., Ding, C.: Fake news detection based on news content and social contexts: a transformer-based approach. Int. J. Data Sci. Anal. (2022). https://doi.org/10.1007/s41060-021-00302-z
42. Hapke, H., Nelson, C.: Building Machine Learning Pipelines, http://oreilly.com/catalog/0636920260912/errata%0Ahttps://github.com/Building-ML-Pipelines/building-machine-learning-pipelines, (2020)
43. Dutta, S., Wei, D., Yueksel, H., Chen, P.Y., Liu, S., Varshney, K.R.: Is there a trade-off between fairness and accuracy? A perspective using mismatched hypothesis testing. In: 37th International Conference on Machine Learning, ICML 2020. pp. 2783–2793 (2020)
44. Kirchner, L., Mattu, S., Larson, J., Angwin, J.: Machine Bias — ProPublica, https://www.propublica.org/article/machine-bias-risk-assessments-in-criminal-sentencing
45. Kleinberg, J., Mullainathan, S., Raghavan, M.: Inherent trade-offs in the fair determination of risk scores. Leibniz Int. Proc. Informatics, LIPIcs. 67, (2017). https://doi.org/10.4230/LIPIcs.ITCS.2017.43
46. Narayanan, A.: Fairness Definitions and Their Politics. In: Tutorial presented at the Conf. on Fairness, Accountability, and Transparency (2018)
47. Dwork, C., Ilvento, C.: Group fairness under composition. In: Proceedings of the 2018 Conference on Fairness, Accountability, and Transparency (FAT* 2018) (2018)
48. Calmon, F.P., Wei, D., Vinzamuri, B., Ramamurthy, K.N., Varshney, K.R.: Optimized pre-processing for discrimination prevention. Adv. Neural Inf. Process. Syst. 2017-Decem, 3993–4002 (2017)
49. Kamishima, T., Akaho, S., Asoh, H., Sakuma, J.: Fairness-aware classifier with prejudice remover regularizer. In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 35–50 (2012)
50. Celis, L.E., Huang, L., Keswani, V., Vishnoi, N.K.: Classification with fairness constraints: A meta-algorithm with provable guarantees. In: Proceedings of the conference on fairness, accountability, and transparency. pp. 319–328 (2019)
51. Agarwal, A., Beygelzimer, A., Dud\'\ik, M., Langford, J., Wallach, H.: A reductions approach to fair classification. In: International Conference on Machine Learning. pp. 60–69 (2018)
52. Udeshi, S., Arora, P., Chattopadhyay, S.: Automated directed fairness testing. In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering. pp. 98–108 (2018)
53. Saleiro, P., Kuester, B., Hinkson, L., London, J., Stevens, A., Anisfeld, A., Rodolfa, K.T., Ghani, R.: Aequitas: A bias and fairness audit toolkit. arXiv Prepr. arXiv1811.05577. (2018)
54. Tan, C., Sun, F., Kong, T., Zhang, W., Yang, C., Liu, C.: A survey on deep transfer learning. In: International conference on artificial neural networks. pp. 270–279 (2018)
55. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: BERT: Pre-training of deep bidirectional transformers for language understanding. arXiv Prepr. arXiv1810.04805. (2018)
56. Li, B., Peng, H., Sainju, R., Yang, J., Yang, L., Liang, Y., Jiang, W., Wang, B., Liu, H., Ding, C.: Detecting Gender Bias in Transformer-based Models: A Case Study on BERT. (2021)
57. Sinha, M., Dasgupta, T.: Determining Subjective Bias in Text through Linguistically Informed Transformer based Multi-Task Network. In: Proceedings of the 30th ACM International Conference on Information \& Knowledge Management. pp. 3418–3422 (2021)
58. Nadeau, D., Sekine, S.: A survey of named entity recognition and classification. Lingvisticae Investig. 30, 3–26 (2007)
59. Excell, E., Moubayed, N. Al: Towards Equal Gender Representation in the Annotations of Toxic Language Detection. (2021)
60. Mishra, S., He, S., Belli, L.: Assessing Demographic Bias in Named Entity Recognition. (2020)
61. Kaneko, M., Bollegala, D.: Unmasking the Mask--Evaluating Social Biases in Masked Language Models. arXiv Prepr. arXiv2104.07496. (2021)
62. Gölz, P., Kahng, A., Procaccia, A.D.: Paradoxes in fair machine learning. Adv. Neural Inf. Process. Syst. 32, (2019)
63. Mastrine, J.: Types of Media Bias and How to Spot It | AllSides, https://www.allsides.com/media-bias/how-to-spot-types-of-media-bias, (2018)
64. NY Times: Don't buy the psuedo-scientific hype

about tornadoes, climate change, https://nypost.com/2021/12/12/dont-buy-the-psuedo-scientific-hype-about-tornadoes-climate-change
65. Murphy, K.P.: Machine learning: a probabilistic perspective. MIT press (2012)
66. Yan, H., Deng, B., Li, X., Qiu, X.: TENER: Adapting Transformer Encoder for Named Entity Recognition. (2019)
67. Yan, H., Qian, T., Xie, L., Chen, S.: Unsupervised cross-lingual model transfer for named entity recognition with contextualized word representations. PLoS One. 16, e0257230 (2021). https://doi.org/10.1371/journal.pone.0257230
68. Torrey, L., Shavlik, J.: Transfer learning. In: Handbook of research on machine learning applications and trends: algorithms, methods, and techniques. pp. 242–264. IGI global (2010)
69. Sinha, K., Jia, R., Hupkes, D., Pineau, J., Williams, A., Kiela, D.: Masked language modeling and the distributional hypothesis: Order word matters pre-training for little. arXiv Prepr. arXiv2104.06644. (2021)
70. BBC: Billie Eilish sorry for mouthing anti-Asian slur in resurfaced video - BBC News, https://www.bbc.com/news/newsbeat-57564878
71. Wu, X., Zhang, T., Zang, L., Han, J., Hu, S.: Mask and infill: Applying masked language model to sentiment transfer. IJCAI Int. Jt. Conf. Artif. Intell. 2019-Augus, 5271–5277 (2019). https://doi.org/10.24963/ijcai.2019/732
72. Barocas, S., Hardt, M., Narayanan, A.: Limitations and Opportunities Solon Barocas , Moritz Hardt , Arvind Narayanan. Fairness Mach. Learn. Limit. Oppotunities. (2019)
73. Google, M.M., Vinodkumar, B., Google Brain, P., Chang, K.-W., Román, V.O.: Bias and Fairness in NLP. (2019)
74. Knights, L.: Fair. Danc. Times. 107, 74–77 (2016). https://doi.org/10.2307/j.ctt1b7x5pt.3
75. Gaucher, D., Friesen, J., Kay, A.C.: Evidence That Gendered Wording in Job Advertisements Exists and Sustains Gender Inequality. J. Pers. Soc. Psychol. 101, 109–128 (2011). https://doi.org/10.1037/a0022530
76. Matfield, K.: Gender Decoder: find subtle bias in job ads, http://gender-decoder.katmatfield.com/, (2016)
77. Wright, R.E.: Logistic regression. (1995)
78. Chang, C.-C., Lin, C.-J.: LIBSVM: a library for support vector machines. ACM Trans. Intell. Syst. Technol. 2, 1–27 (2011)
79. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., Bengio, Y.: Generative adversarial nets. Adv. Neural Inf. Process. Syst. 27, (2014)
80. Bobko, P., Roth, P.L.: The four-fifths rule for assessing adverse impact: An arithmetic, intuitive, and logical analysis of the rule and implications for future research and practice. In: Research in personnel and human resources management. Emerald Group Publishing Limited (2004)
81. IBM Cloud Paks: Fairness metrics overview - IBM Documentation, https://www.ibm.com/docs/en/cloud-paks/cp-data/4.0?topic=openscale-fairness-metrics-overview
82. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. In: Advances in neural information processing systems. pp. 5998–6008 (2017)
83. Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Passos, A., Cournapeau, D., Brucher, M., Perrot, M., Duchesnay, E.: Scikit-learn: Machine Learning in {P}ython. J. Mach. Learn. Res. 12, 2825–2830 (2011)
84. Ginting, P.S.B., Irawan, B., Setianingsih, C.: Hate speech detection on twitter using multinomial logistic regression classification method. In: 2019 IEEE International Conference on Internet of Things and Intelligence System (IoTaIS). pp. 105–111 (2019)
85. Wang, Y., Zhou, Z., Jin, S., Liu, D., Lu, M.: Comparisons and selections of features and classifiers for short text classification. In: IOP Conference Series: Materials Science and Engineering. p. 12018 (2017)
86. Ali, J., Khan, R., Ahmad, N., Maqsood, I.: Random forests and decision trees. Int. J. Comput. Sci. Issues. 9, 272 (2012)
87. Fauzi, M.A.: Random Forest Approach fo Sentiment Analysis in Indonesian. Indones. J. Electr. Eng. Comput. Sci. 12, 46–50 (2018)
88. Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Ann. Stat. 1189–1232 (2001)
89. Peters, M.E., Neumann, M., Iyyer, M., Gardner, M., Clark, C., Lee, K., Zettlemoyer, L.: Deep contextualized word representations, (2018)
90. Windeatt, T.: Accuracy/diversity and ensemble MLP classifier design. IEEE Trans. Neural Networks. 17, 1194–1211 (2006)
91. Wang, Y., Sohn, S., Liu, S., Shen, F., Wang, L., Atkinson, E.J., Amin, S., Liu, H.: A clinical text classification paradigm using weak supervision and deep representation. BMC Med. Inform. Decis. Mak. 19, 1–13 (2019). https://doi.org/10.1186/s12911-018-0723-6
92. Sun, C., Qiu, X., Xu, Y., Huang, X.: How to Fine-Tune BERT for Text Classification? In: Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics). pp. 194–206 (2019)
93. Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., Stoyanov, V.: RoBERTa: A Robustly Optimized BERT Pretraining Approach. 2383–2392 (2019)
94. Schick, T., Schütze, H.: Exploiting cloze questions for few shot text classification and natural language inference. arXiv Prepr. arXiv2001.07676. (2020)
95. Sanh, V., Debut, L., Chaumond, J., Wolf, T.:

DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv Prepr. arXiv1910.01108. (2019)
96. Dacrema, M.F., Cremonesi, P., Jannach, D.: Are we really making much progress? A worrying analysis of recent neural recommendation approaches. In: Proceedings of the 13th ACM Conference on Recommender Systems. pp. 101–109 (2019)
97. Chouldechova, A.: Fair prediction with disparate impact: A study of bias in recidivism prediction instruments. Big data. 5, 153–163 (2017)
98. MacCarthy, M.: Standards of fairness for disparate impact assessment of big data algorithms. Cumb. L. Rev. 48, 67 (2017)
99. Beutel, A., Chen, J., Zhao, Z., Chi, E.H.: Data decisions and theoretical implications when adversarially learning fair representations. arXiv Prepr. arXiv1707.00075. (2017)
100. Garg, P., Villasenor, J., Foggo, V.: Fairness metrics: A comparative analysis. In: 2020 IEEE International Conference on Big Data (Big Data). pp. 3662–3666 (2020)