HINTER: Exposing Hidden Intersectional Bias in Large Language Models

Anonymous TACL submission

Abstract

We propose HINTER, a test technique that synergistically combines *mutation analysis*, *dependency parsing* and *metamorphic oracles* to automatically detect *intersectional bias* in LLMs. HINTER *generates test inputs* by systematically mutating sentences using *multiple mutations*, *validates* inputs via a *dependency invariant* checker and *detects biases* by checking the LLM response on the original and mutated sentences. We evaluate HINTER using six LLM architectures and 18 LLM models (GPT3.5, Llama2, BERT, etc) and find that it can expose intersectional bias in 14.61% of the inputs it generates. We also find that the use of the dependency invariant checker reduces the incorrect inputs by an order of magnitude. Our results also show that 16.62% of intersectional bias errors are *hidden*, meaning that their corresponding atomic cases do not trigger the biases, thus, emphasizing the need for intersectional bias testing.

1 Introduction

Large Language Models (LLMs) have become vital components of critical services and products in our society (Zhao et al., 2023). LLMs are increasingly adopted in critical domains inluding NLP, legal, health and programming tasks (Liang et al.). For instance, popular pre-trained models (e.g., BERT, GPT and Llama) have been deployed across multiple NLP, legal and coding tasks (Chalkidis et al., 2022; Angwin et al., 2019).

Despite their popularity and effectiveness across various tasks, LLMs face the risk of portraying bias towards specific individuals (Mehrabi et al., 2021). Discrimination in LLMs may have serious consequences e.g., miscarriage of justice in law enforcement (Angwin et al., 2019).

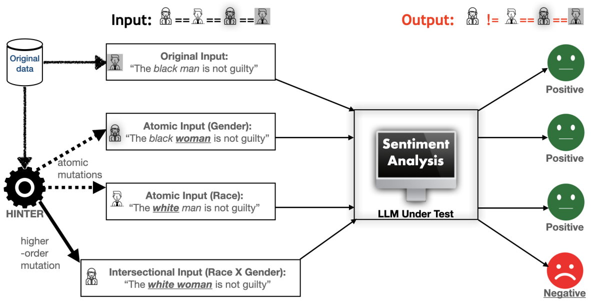

Input: == == ==
Output: != == ==
Original data
Original Input: "The *black man* is not guilty"
Atomic Input (Gender): "The *black* ***woman*** is not guilty"
Atomic Input (Race): "The ***white*** *man* is not guilty"
Intersectional Input (Race X Gender): "The ***white woman*** is not guilty"
atomic mutations
HINTER
higher -order mutation
Sentiment Analysis
LLM Under Test
Positive
Positive
Positive
Negative

Figure 1: HINTER exposes a hidden intersectional bias

To this end, we pose the following question: *How can we systematically discover intersectional (i.e., non-atomic) bias in LLMs?* As illustrated in Figure 1 and exemplified in Table 1, we aim to *automatically generate valid test inputs that expose intersectional bias*. We focus on intersectional bias relating to individuals (rather than groups) since individual fairness is more fine-grained and captures the subtle unfairness relating to specific intersectional individuals. In contrast, group fairness is more coarse-grained and it often conceals individual differences within groups.

Discovering intersectional bias is challenging due to its complex nature and the huge number of possible combinations of attributes involved. Concretely, the three major technical challenges involved are: (1) **test generation**– systematically generate inputs that are likely to uncover intersectional bias; (2) **test validity** – automated test input validation w.r.t. to original dataset; (3) **bias detection** – detect (hidden) intersectional bias at scale.

We propose HINTER[1], an automated black-box technique which synergistically combines *mutation analysis*, *dependency parsing* and *metamorphic oracles* to expose *intersectional* bias (*see* Figure 2). HINTER **generates test inputs** by systematically mutating sentences via **higher-order mu-**

[1]HINTER refers to (1) a tool to "***hint***" (discover) **intersectional** bias; and (2) it is the German word for "*behind*", implying that intersectional bias is hidden *behind* atomic bias.

tation. Next, it **validates** inputs via a **dependency invariant** which checks that the **parse trees** of the generated input and the original text are similar. Then, HINTER **detects bias** by comparing the LLM model outcome of the original text and the generated input(s). Using 18 LLMs, we demonstrate the performance of HINTER in exposing (hidden) intersectional bias. We also investigate *whether it is necessary to perform intersectional bias testing, despite atomic bias testing.*

This paper makes the following contributions:

- We propose an **automated intersectional bias testing technique (HINTER)** that employs a combination of *multiple mutations, dependency parsing and metamorphic oracles*.
- **Evaluation:** We conduct an empirical study to examine the effectiveness of HINTER using three sensitive attributes, five datasets, six LLM architectures, five tasks, and 18 models (GPT3.5, Llama2, BERT, etc).
- **Atomic vs. Intersectional Bias:** We **demonstrate the need for intersectional bias testing** by showing that 16.62% of intersectional bias errors are *hidden*, i.e., their corresponding atomic tests do not trigger biases.

2 Background

Motivating Example: Figure 1 illustrates a case of the hidden intersectional bias we aim at. Given a dataset (containing e.g., 1st sentence in Figure 1), we generate higher-order mutants that expose intersectional bias in the LLM under test. In this example, the higher-order mutant (4th sentence) exposes a *hidden* bias since its corresponding atomic mutations (2nd and 3rd sentences) do not induce any bias. Table 1 shows some instances of *hidden intersectional bias* found by HINTER. These examples show how intersectional bias is *hidden* during atomic bias testing and demonstrates the need for intersectional bias testing. We provide a webpage to test these examples.[2]

HINTER Overview: Figure 2 illustrates the workflow of HINTER. Given an existing dataset (e.g., training dataset), HINTER first *generates bias-prone inputs* by employing higher-order mutations. It mutates a sentence using word pairs extracted from SBIC (Sap et al., 2020), a well-known bias corpus containing 150K social media posts covering a thousand demographic groups[3]. Second, HINTER ensures that the generated input are *valid* by checking that it has a similar dependency parse tree with the original text (*see* Figure 3). Finally, HINTER's metamorphic oracle discovers intersectional bias by comparing the LLM outcome of the original text and the generated input[4]. It further detects a *hidden* intersectional bias by comparing the outcomes of atomic and intersectional mutants.

Problem Formulation: Given an LLM f, HINTER aims to determine whether the inputs relating to individuals characterized by multiple sensitive attributes face bias. Thus, it generates a test suite T that includes individuals associated with multiple sensitive attributes.

Consider Figure 1 that is focused on two sensitive attributes, "`race`" and "`gender`". HINTER produces an intersectional bias test suite (T_{RXG}) based on the original dataset ($Orig$) by simultaneously mutating words associated with each attribute (e.g., "*black*" to "*white*" and "*man*" to "*woman*" for attributes "`race`" and "`gender`" respectively). We also produce an atomic bias test suite (T_R) by mutating *only* a single attribute (e.g., "white" to "black" for "race" *only*) at a time. The model outcome ($O_t = f(t)$) for every test input ($t \in T_{RXG}$) characterizes how the LLM captures an intersectional individual t for model f.

This setting allows to determine the following:

(a) ***Atomic Bias*** (aka individual bias) occurs when the LLM model outcomes (O) for two individuals $\{t1, t2\}$ are different (($O_{t1} \neq O_{t2}$) even though both individual $t1$ and $t2$ are similar for the task at hand, but only differ by *exactly* one sensitive attributes (e.g., `gender`). For instance, since the *first* and *second* inputs in Figure 1 only differ in *gender* attribute, we say there is an *atomic* bias if their LLM outcomes differ. Our definition of atomic bias is in line with previous literature on fairness metrics (Dwork et al., 2012; Friedman and Nissenbaum, 1996; Verma and Rubin, 2018; Crawford, 2017; Narayanan, 2018). These works require that *similar individuals* should be *treated similarly*.

(b) Intersectional Bias occurs when the LLM model outcomes (O) for two individuals $\{t1, t2\}$ are different (($O_{t1} \neq O_{t2}$) even though $t1$ and $t2$ are similar for the task at hand, and only differ by *at least* two sensitive attributes (e.g., `race` and `gender`). For instance, the *first* and *fourth* test in-

[2]https://huggingface.co/spaces/Anonymous1925/Hinter
[3]https://maartensap.com/social-bias-frames/

[4]We say a bias (or error) is discovered when the outcomes are different and the input is bias-inducing. Otherwise the generated input is benign, and there is no bias (no error).

Table 1: Examples of Hidden Intersectional Bias Errors found by HINTER in GPT3.5 using the IMDB dataset. Mutations are marked in **bold text and underlined**. Bias (✘) is characterized by changes in GPT3.5 predictions. ("-" = Not Applicable, ✔ = No Bias/Benign, ✘ = Bias detected)

ID	Sensitive Attributes	Mutations	Test/Mutant Type	Test Cases Exposing Hidden Intersectional Bias	GPT3.5 Prediction	Bias
19375	-	-	Original Input	"... There is a special heaven reserved for **people** who make the world laugh . .. **British** moviegoers will recognise the fat one from Cannon and Ball ... "	Negative	-
	Race	British-> Pakistani	**Atomic Input I**	"... There is a special heaven reserved for **people** who make the world laugh . .. **Pakistani** moviegoers will recognise the fat one from Cannon and Ball ... "	Negative	✔
	Gender	people->trans women	**Atomic Input II**	"... There is a special heaven reserved for **trans women** who make the world laugh . .. **British** moviegoers will recognise the fat one from Cannon and Ball ... "	Negative	✔
	Race & Gender	people->trans women British-> Pakistani	**Intersectional Input**	"... There is a special heaven reserved for **trans women** who make the world laugh ... **Pakistani** moviegoers will recognise the fat one from Cannon and Ball ...	**Positive**	✘
44660	-	-	Original Input	"The Aristorcats is a hilarious film that not many **people** have seen. ... I think that The Aristorcats is so funny and **cute**. ..."	Positive	-
	Gender	people->non-masculine people	**Atomic Input I**	"The Aristorcats is a hilarious film that not many **non-masculine people** have seen. ... I think that The Aristorcats is so funny and **cute**. ..."	Positive	✔
	Body	cute->manly	**Atomic Input II**	"The Aristorcats is a hilarious film that not many **people** have seen. ... I think that The Aristorcats is so funny and **manly**. ..."	Positive	✔
	Gender & Body	people->non-masculine people, cute->manly	**Intersectional Input**	"The Aristorcats is a hilarious film that not many **non-masculine people** have seen. ... I think that The Aristorcats is so funny and **manly**. ..."	**Negative**	✘

puts in Figure 1 are two such inputs ($t1$ and $t2$). This definition is in line with the ML literature on **intersectional bias** (Buolamwini and Gebru, 2018; D'ignazio and Klein, 2020).

(c) Hidden Intersectional Bias refers to when atomic inputs ($atomic_r \in T_R$, $atomic_g \in T_G$) characterized by the test suites (T_R, T_G) are benign (trigger no bias) but their corresponding intersectional test input ($inter_{rXg} \in T_{RXG}$) from the test suites (T_{RXG}) trigger a bias. In particular, we say there is a *hidden intersectional bias* when the LLM model outcomes (O) for the atomic test inputs ($atomic_r$, $atomic_g$) are similar, but their corresponding intersectional test input ($inter_{rXg}$) produce a different outcome – ($O_{atomic_g} == O_{atomic_r}$) ≠ ($O_{inter_{rXg}}$). The *second* and *third* test inputs in Figure 1 are two such atomic inputs, and the *fourth* input is their corresponding intersectional input.

(d) Atomic vs. Intersectional Bias: We determine the differences between intersectional bias and atomic bias testing by inspecting the outcomes of our test suites ($O(T_{RXG})$ vs $O(T_R)$ vs $O(T_G)$). We inspect whether original inputs and mutations producing the atomic test suites (T_R or T_G) trigger the same biases as the intersectional bias test suites (T_{RXG}).

3 HINTER Approach

Figure 2 illustrates the workflow of our approach. Given, as *inputs*, an existing dataset, sensitive attribute(s) and bias dictionary, HINTER generates a bias-prone test suite. The *output* of HINTER is the *bias-prone test suite* which contains mutations of the original dataset and *intersectional bias instances* it detects.

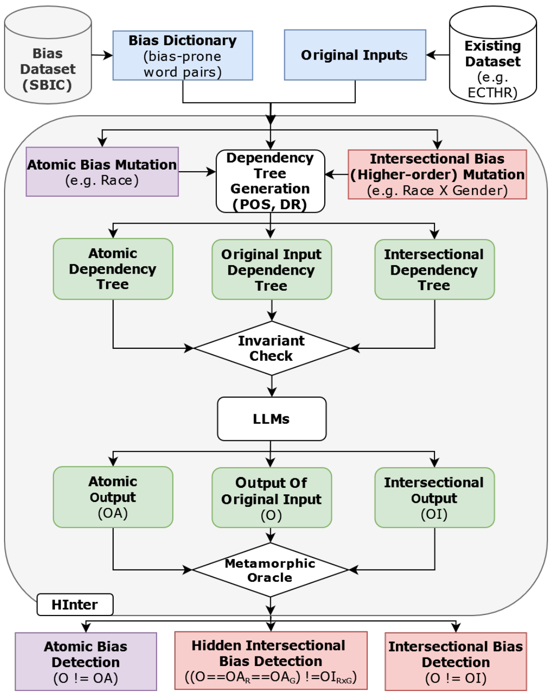

Bias Dataset (SBIC)
Bias Dictionary (bias-prone word pairs)
Original Inputs
Existing Dataset (e.g. ECTHR)
Atomic Bias Mutation (e.g. Race)
Dependency Tree Generation (POS, DR)
Intersectional Bias (Higher-order) Mutation (e.g. Race X Gender)
Atomic Dependency Tree
Original Input Dependency Tree
Intersectional Dependency Tree
Invariant Check
LLMs
Atomic Output (OA)
Output Of Original Input (O)
Intersectional Output (OI)
Metamorphic Oracle
HInter
Atomic Bias Detection (O != OA)
Hidden Intersectional Bias Detection ((O==OA_R==OA_G) !=OI_RxG)
Intersectional Bias Detection (O != OI)

Figure 2: Workflow of HINTER

HINTER involves the following *three main steps*:

Step 1: Higher-order Mutation: HINTER transforms each original input into a bias-prone mutant via higher-order mutation. It replaces word(s) in the original input with sensitive words using the bias-prone dictionary. Our bias dictionary is automatically extracted from the SBIC corpus (Sap et al., 2020). It contains a set of bias-prone word-pairs for each sensitive attribute, e.g., the pair ("man":"woman") mapped to the *gender* attribute.

Algorithm 1 presents our intersectional bias testing algorithm. Given an original dataset C, sensitive attributes $S1, S2$ and a bias dictionary P, HINTER generates a test suite (T). Each test is a mutants (m) of the original text c using word

Algorithm 1 Intersectional Bias testing

```
1: {Input: Dataset $C$, LLM $M$, Dictionary $P$, Attributes $S1$, $S2$}
2: {Output: test suite $T$, intersectional bias $E$ & hidden bias $H$}
3: Function HINTER($C$, $M$, $P$, $S1$, $S2$)
4: $T_{list}$ = [], $E_{list}$ = [], $H_{list}$ = []
5: {Test suite $T$, intersectional bias $E$ & hidden bias $H$ lists}
6: for $c$ in $C$ do
7:     $O[c] \leftarrow M(c)$ {Store LLM outcome for the original text}
8:     $P_1 = P[S1], P_2 = P[S2]$
9:     {Get word pairs for attributes $S1, S2$ from dictionary $P$}
10:    for $p_1$ in $P_1$ do
11:        for $p_2$ in $P_2$ do
12:            if $p_1[0], p_2[0]$ in $c$ then
13:                $t_1 \leftarrow c.replace(p_1[0], p_1[1])$
14:                {First Atomic Mutation using Bias list $P_1$}
15:                $t_2 \leftarrow c.replace(p_2[0], p_2[1])$
16:                {Second Atomic Mutation using Bias list $P_2$}
17:                $m \leftarrow t1.replace(p_2[0], p_2[1])$
18:                {Intersectional (Higher-order) Mutation}
19:                if $InvCheck(c, m)$ then
20:                    {Dependency invariant check}
21:                    $T_{list} \cup m$ {Store mutant in test suite}
22:                    if $M(m) \neq O[c]$ then
23:                        {Metamorphic Test Oracle}
24:                        $E_{list} \leftarrow E_{list} \cup (m, c, M(m), O[c])$
25:                        {Store bias inducing mutant}
26:                        if $InvCheck(c, t_1)$ then
27:                            if $InvCheck(c, t_2)$ then
28:                                if $M(t_1) \equiv O[c] \equiv M(t_2)$ then
29:                                    $H_{list} \cup m$ {Store hidden bias}
30: return $T_{list}, E_{list}, H_{list}$
```

pairs (P_1 and P_2) from dictionary P. To mutate the original input, HINTER searches the original text for the presence of any bias-prone word pairs using the sensitive attribute(s) at hand (lines 7-11). If it finds bias-prone words (line 12), then it mutates the word by replacing it with its pairs (lines 13-18). For intersectional bias testing, HINTER needs to find at least two bias-prone words in the original text belonging to two different attributes (line 12)[5]. Then, HINTER performs higher-order mutation (i.e., two mutations simultaneously) for the two sensitive attributes (lines 13 and 18).

Figure 1 shows a case of intersectional bias testing for *race X gender*. HINTER first searches the original text for bias-prone words associated to the two sensitive attributes (words that exist in the dictionary), then it identifies the words "black" and "man" and constructs all possible pairs of these words from the bias dictionary. Lets assume that it finds only two pairs in the dictionary – {"black":"white"} and {"man":"woman"}, therefore it constructs one mutation, the pair "white woman" (the last mutant in Figure 1). HINTER also performs atomic mutation by performing only one replacement per attribute at a time, e.g., performing atomic mutations for only "race" results in the text "white man", the third sentence in Figure 1.

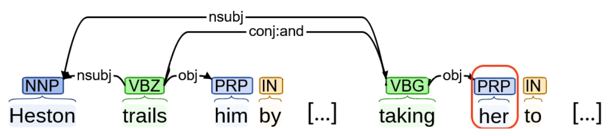
[5]For atomic bias testing, this is relaxed to finding a single instance of bias-prone words for the attribute at hand.

Algorithm 2 Dependency Invariant Check

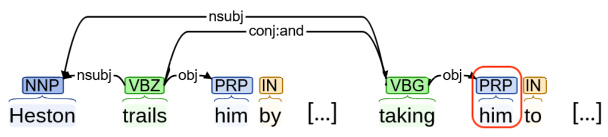
```
1: {Input: Original Text $c$, Mutant $m$ generated by HINTER }
2: {Output: Are the POS & DP of $c$ & $m$ similar: true or false}
3: Function InvCheck(c, m)
4: $cs \leftarrow sentenceSplit(c), ms \leftarrow sentenceSplit(m)$
5: {Split $c$ & $m$ into sentences}
6: if size($cs$) ≠ size($ms$) then
7:     {Compare the number of sentences}
8:     return false
9: for $i$ from 1 to size($cs$) do
10:    {Loop for each sentence}
11:    $cdt$ ← dependencyTree($cs[i]$)
12:    $mdt$ ← dependencyTree($ms[i]$)
13:    {Compute the dependency trees}
14:    $similarPOS$ ← tolerantTableComp($cdt.pos, mdt.pos$)
15:    {Compare POS of mutant and original sentences}
16:    if $similarPOS \neq$ true then
17:        return false
18:    $similarDP$ ← tolerantTableComp($cdt.dep, mdt.dep$)
19:    {Compare DP of mutant and original sentences}
20:    if $similarDP \neq$ true then
21:        return false
22: return true
```

Step 2: Dependency Invariant Check: The parse tree details the structure, POS and dependency relations among words in the text (e.g., Figure 3). The main goal of this step is to check that the following **invariant** holds – *the parts of speech (POS) and dependency relations (DR) of the generated inputs are similar to the original input.* The intuition is that the generated inputs need to have a similar grammatical structure with the original input to preserve the intention and semantics of the original text. We apply invariant checking (`InvCheck`) on the both original and generated inputs (e.g., lines 19, 26 and 27 of algorithm 1).

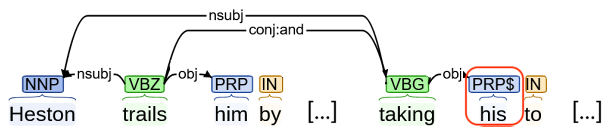
Algorithm 2 details the dependency invariant checker. `InvCheck` first splits each text into sentences (line 4) and confirms that both texts have the same number of sentences (line 6). For each sentence, it generates the parse trees for the original input and the generated test input (lines 11-12) and checks that the parse trees of both inputs are similar (lines 14-21). It checks that each sentence in the generated input conforms to the parts of speech (POS) and dependency relations (DR) of its corresponding sentence in the original input (lines 14 and 16). This comparison is performed using `tolerantTableComp` (*see* algorithm details in the appendix). For instance, the `tolerantTableComp` accounts for possible insertions made during the mutation since some replacements may insert additional words, e.g., for the *Body* attribute, the word "*man*" may be replaced with "*disabled man*". To ac-



(a) Original Input



(b) Valid Input generated by HINTER



(c) Invalid Input discarded by HINTER's dependency invariant

Figure 3: Dependency invariant check. Valid Input (b) conforms to the original text (a), while discarded input (c) does not – "`PRP$ (his)`" `!=` "`PRP (him)`").

commodate for such insertions, the invariant check uses the `tolerantTableComp` to compare the two-word sequences in both the original and generated text within a tolerance defined by their absolute size difference. This ensures that the number of unmatched elements are within the allowable limit (absolute size difference) of both inputs.

HINTER *discards* inputs whose parse trees are *nonconforming* to the parse tree of the original inputs, i.e., fail our dependency invariant check. However, if the parse tree of the generated input (i.e., mutant) conforms with that of the original input, then HINTER adds such inputs to the resulting test suite (line 21 of algorithm 1). We refer to such inputs, that *pass* the dependency invariant check, as valid/generated inputs.

Figure 3 shows two mutants, a valid mutant/input ((b) and an invalid/discarded input (c)) generated from the original input (a). The *first* mutant (Figure 3(b)) *conforms* to the dependency structure of the original input (Figure 3(a)). Thus, it passes HINTER's dependency invariant check and it is added to the test suite as a valid/generated input. However, the *second* mutant (Figure 3(c)) is *non-conforming* to the the dependency structure of the original input (Figure 3(a)) since "`PRP$ (his)`" `!=` "`PRP (him)`". Hence, the mutant Figure 3(c) fails the dependency invariant check of HINTER and it is *not* added to the resulting bias test suite, thus it is called a *discarded* input.

Step 3: Metamorphic Test Oracle: Figure 2 illustrates how HINTER detects intersectional bias. After the invariant check (**Step 2**), HINTER (algorithm 1) feeds the generated test input and its corresponding original input (line 7) into the LLM under test separately and compares the LLM outputs (line 22). HINTER detects intersectional bias (aka error), when an intersectional mutant produces a different model outcome from the original test input (line 24). Concretely, it employs a metamorphic test oracle to detect bias by checking if the model outcome *changes* (is *different*) between the generated mutant and the original input (line 22). Otherwise, when the model outcomes remains unchanged (remains the same), we say the test input is *benign* and it does not induce a bias. The **metamorphic property** leveraged in this test oracle is that *the LLM outcome for the original input and the generated input (mutant) should remain the same since both inputs are logically similar for the task at hand.* Consider the original input and generated mutants in Figure 1, these inputs are similar in the context of the sentiment analysis task. Consequently, all four inputs should produce the same outcome, i.e., a "positive" label/prediction. We say that a *hidden intersectional bias* is detected (lines 22, 28-29 of algorithm 1) when there is an intersectional bias but its corresponding atomic mutants are not exposing any bias. Figure 2 illustrates a hidden intersectional bias workflow.

Metamorphic oracle detects intersectional bias when the LLM outcome of the original input (first sentence) differs ("positive" != "negative") from that of the intersectional mutant (fourth sentence), e.g., Figure 1. An atomic bias is *not* detected in this example since the LLM outcome for the original input (first sentence) and the atomic mutants (the second and third sentences) are the same ("positive"). We refer to the last/fourth mutant in Figure 1 as a *hidden* intersectional bias because its corresponding atomic mutations do not expose bias(es).

4 Experimental Setup

Research Questions:

RQ1* HINTER*'s Effectiveness: How effective is HINTER in discovering intersectional bias?

RQ2 Grammatical Validity: Are the inputs generated by HINTER grammatically valid?

RQ3 Dependency Invariant: What is the contribution of HINTER's dependency invariant check?

RQ4 Atomic Bias vs. Intersectional Bias: Do intersectional bias-inducing mutations and original texts *also* reveal atomic bias, and vice versa?

Tasks and Datasets: Table 2 details our experimental datasets/tasks. We have chosen these datasets due to their critical nature, availability of LLM models and popularity. These datasets span different tasks and are complex, e.g., long texts and multi-class/-label classification.

LLM Architectures and Models: Our experiments involve 18 LLM models. We employ six LLM architectures in our experiments covering the three main pre-trained model architectures – encoder-only (e.g., BERT, Legal-BERT, RoBERTa), decoder-only, (GPT.3.5, Llama) and encoder-decoder (DeBERTa) architectures. We have chosen these LLM models to ensure varying model settings. Due to space limitations, we provide model details (e.g., accuracy) in our supplementary material and artifact.[6]

Sensitive Attributes: This work employs three well-known sensitive attributes, namely "*race*", "*gender*" and "*body*". For atomic bias, we consider each attribute in isolation. For intersectional bias, we simultaenously combine every two attributes (i.e., $N = 2$) namely "*race X gender*", "*body X gender*" and "*body X race*". We have employed (these) three senstive attributes due to their popularity and the prevalence of attributes in available datasets[7]. Figure 1 and Table 1 illustrate these attributes.

Metrics and Measures: We report the following metrics in our experiments:

1. Bias Error Rate is the proportion of generated test inputs that induce bias (*see* **RQ1**).

2.) Proportion of Bias-inducing Original Text: We compute the proportion of distinct original texts that lead to bias (*see* **RQ1** and **RQ4**).

3.) Grammatical Validity Scores is the grammatical correctness scores reported by Grammarly (Hoover et al., 2009) for original, generated, and discarded inputs (*see* **RQ2** and **RQ3**).

4.) Valid Mutants and Discarded Mutants: We report the number and proportion of mutants that *pass* (aka *valid mutants*) our dependency invariant check or *fail* (aka *discarded mutants*) our dependency invariant check (*see* **RQ3**).

5.) Hidden Intersectional Bias Error Rate: We compute the proportion of intersectional bias errors that are *hidden*, i.e., their corresponding atomic mutations are benign (*see* **RQ1**/Table 2).

6.) Bias-inducing Original inputs and Mutations: We report the number of mutations and original inputs that induce bias (*see* **RQ4**).

[6]https://github.com/Anonymous1925/Hinter

[7]Note that most (69%, nine out of 13) datasets have at most three attributes (Gohar and Cheng, 2023).

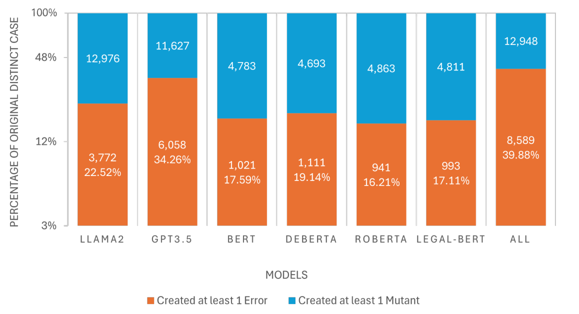


Figure 4: Distinct Intersectional bias-inducing original inputs

7.) False Positives: The number of bias induced by discarded/invalid mutants, i.e., inputs that *fail* dependency invariant check (*see* **RQ3**)[8].

Bias dictionary: We automatically extracted it from the "Social Bias Inference Corpus" (SBIC) (Sap et al., 2020) which contains 150K structured annotations of social media posts, covering a thousand demographic groups. Our dictionary contains curated semantically meaningful bias alternatives totalling 116 words for "race", 230 words for "gender" and 98 words for "body" attributes. We also manually inspected and validated the list of word pairs for each attribute.

Prompt Configuration: We employ few-shot prompting to query the pre-trained models (GPT3.5 and Llama2). Few-shot prompting performed best in our preliminary evaluation, e.g., in comparison to zero-shot prompting. The max prompt/input size of the fine-tuned models is 512 tokens, while the two pre-trained models (GPT3.5 and Llama2) support up to 4096 tokens. To mitigate model hallucination and randomness, we set the temperature for each query to zero for Llama2. We also turn off sampling and consistently use the same examples to ensure results as deterministic as possible. More details and examples of our prompts are provided in the supplementary material, artifact and huggingface.[9]

Implementation Details and Platform: HINTER and our data analysis were implemented in about 6 KLOC of Python. All experiments were conducted on a compute server with four Nvidia Tesla V100 SXM2 GPUs, two Intel Xeon Gold 6132 processors (2.6 GHz), and 768 GB of RAM.

Table 2: Details of Intersectional Bias Error Rates found by HINTER. "Ⓘ" means "Intersectional Bias Error Rate" and "Ⓗ" means "Hidden Intersectional Bias Error Rate"

Datasets (Domain)	**Tasks**	**Class.**	**#labels**		**Llama 2**	**GPT 3.5**	**Legal-BERT**	**BERT**	**De-BERTa**	**Ro-BERTa**	**Total**
ECtHR	Judgment	Multi-Label	10+1	Ⓘ	51.74	59.03	5.02	4.65	5.10	3.75	11.37
(ECHR)	Prediction			Ⓗ	(13.24)	(13.70)	(45.12)	(15.35)	(17.26)	(18.18)	(18.06)
LEDGAR	Contract	Multi-Class	100	Ⓘ	25.35	7.04	0.00	0.00	0.00	1.05	4.58
(Contracts)	Classif.			Ⓗ	(12.37)	(0.00)	(0.00)	(0.00)	(0.00)	(0.00)	(12.24)
SCOTUS	Issue Area	Multi-Class	14	Ⓘ	12.32	21.32	1.02	2.34	4.76	2.25	10.60
(US Law)	Prediction			Ⓗ	(26.33)	(19.60)	(40.87)	(13.20)	(5.86)	(8.30)	(18.92)
EURLEX	Document	Multi-Label	100	Ⓘ	89.62	91.00	19.99	20.28	19.02	22.97	21.66
(EU Law)	Prediction			Ⓗ	(1.35)	(0.49)	(37.53)	(7.46)	(11.86)	(5.62)	(14.73)
IMDB	Sentiment	Binary	2	Ⓘ	3.52	7.12	N/A	N/A	N/A	N/A	5.43
(Reviews)	Analysis			Ⓗ	(28.38)	(28.03)	N/A	N/A	N/A	N/A	(28.13)
All	**All**	-	-	Ⓘ	12.64	17.26	14.11	14.22	13.69	15.57	14.61
				Ⓗ	(16.99)	(18.19)	(38.46)	(8.36)	(12.45)	(6.64)	16.62

5 Results

RQ1 HINTER's Effectiveness: This experiment investigates the intersectional bias error rate exposed by HINTER, the proportion of *hidden* intersectional bias and the proportion of original inputs that induce *at least* one intersectional bias when mutated. This experiment involves all five datasets and 18 LLM models – all models for the four legal datasets, and the two (2) pre-trained LLMs (GPT3.5 and Llama2) for the IMDB dataset[10]. Table 2 and Figure 4 present our findings.

Results: We found that *about one in seven (14.61%) inputs generated by* HINTER *revealed intersectional bias.* Table 2 shows that 16.62% of intersectional bias found by HINTER are *hidden*, i.e., their corresponding atomic test inputs do not trigger bias and shows that HINTER exposed intersectional bias with 12.64% to 17.26% error rate across all models. GPT3.5 has the highest (17.26%) error rate, while the Llama2 model has the lowest (12.64%) error rate. Inspecting the proportion of distinct original inputs that lead to intersectional bias, we observed that *about* two in five (39.88%) original inputs lead to at least one intersectional bias when mutated by HINTER (*see* Figure 4). Overall, results show that HINTER is effective in exposing (hidden) intersectional bias.

HINTER is effective in exposing (hidden) intersectional bias: 14.61% of the inputs generated by HINTER exposed intersectional bias, and 16.62% of the intersectional bias errors exposed by HINTER are hidden.

[8] Users are saddled with false positives, i.e., cases that do not conform to the parse tree of the original input.

[9] https://huggingface.co/spaces/Anonymous1925/Hinter

[10] The 16 fine-tuned LLMs are inappropriate for IMDB, since they are specifically fine-tuned for the 4 legal datasets.

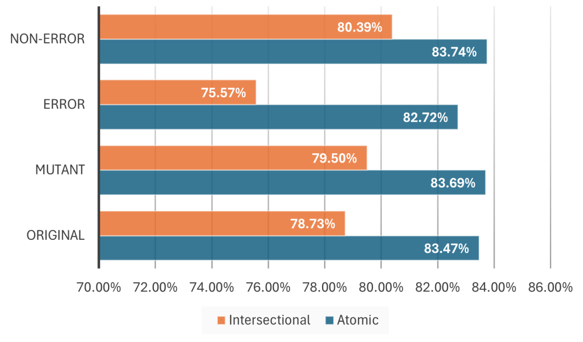


Figure 5: Grammatical Validity of the inputs generated by HINTER, showing Grammarly's weighted mean score for the original (human-written) text vs the HINTER's generated inputs (mutants).

RQ2 Grammatical Validity: This experiment evaluates the *grammatical validity* of the inputs generated by HINTER in comparison to (human-written) original texts. We randomly sampled inputs from the legal datasets[11] and fed them to Grammarly (Hoover et al., 2009)[12] to compute their grammatical validity. This experiment involves about 68K mutants for atomic and intersectional bias testing and their (about 10K) corresponding original inputs. Our sampling aims to balance the set of mutants and originals across sensitive attribute, datasets and mutation type (i.e., atomic and intersectional.) Figure 5 presents our results.

Results: We observed that *the inputs generated by* HINTER *are as valid as the original human-written text.* Figure 5 shows that *the inputs generated by* HINTER *have similar grammatical scores as the original (human-written) text* for both atomic and intersectional bias testing campaigns. For in-

[11] We employ the legal dataset due to their complexity and longer text in comparison to the IMDB dataset.

[12] Grammarly is a popular tool for checking the spelling, grammar and errors in English text.

tersectional bias testing, the weighted mean score of the original text (78.73%) is similar to that of the inputs generated by HINTER (79.50%). We also found that *the grammatical validity of error-inducing inputs is slightly lower than that of benign, non-error-inducing inputs* for both the original inputs and HINTER's generated input (e.g., 82.72 vs 83.74 for atomic bias testing). The Grammarly scores for generated inputs (mutants) remain similar to their corresponding original inputs (*see* Figure 5). These results show that HINTER generates grammatically valid inputs and preserves the grammatical validity of the original input.

HINTER *preserves the grammatical validity of the original (human-written) inputs:* HINTER*'s generated inputs are as grammatically valid as the original (human-written) texts.*

RQ3 Dependency Invariant

RQ3.1 Number of Valid vs. Discarded Inputs: We examine the proportion of mutants that pass or fail HINTER's invariant check, it involves all mutants (valid and discarded test inputs) generated by HINTER for all datasets. Figure 6 presents our findings.

Results: We found that HINTER*'s invariant check is effective in identifying invalid mutated inputs:* HINTER *discards the majority (97.58%) of mutated inputs as invalid because their dependency parse tree does not align with the parse tree of the reference original text.* Figure 6 shows that most input mutations (129M/132M) result in a different dependency parse tree in comparison to the original text. Intuitively, this means that the majority of generic mutations create inputs that have a different meaning or intent in comparison to the original reference text. We observed that *only* 2.42% of input mutations *pass* our invariant check. This suggests that generic mutations (without our invariant check) mostly result in invalid test inputs.[13] Intersectional bias mutations and longer texts are more likely to be invalid (fail our dependency invariant) due to their complexity. These results emphasize the importance of HINTER's invariant check.

HINTER*'s invariant check effectively identifies invalid inputs: It discarded millions (129M/132M) of mutants whose dependency parse trees do not conform to the original text.*

[13]Generic mutation-based bias testing approaches (e.g., MT-NLP (Ma et al., 2020)) mutate inputs without a validity check.

RQ3.2 False Positives: We investigate the false positives (fake errors) developers would have been saddled with inspecting *without* HINTER's invariant check. This experiment involves the valid inputs, discarded inputs and their corresponding errors for the 16 fined-tuned models using the four legal datasets.[14] Figure 7 presents our results.

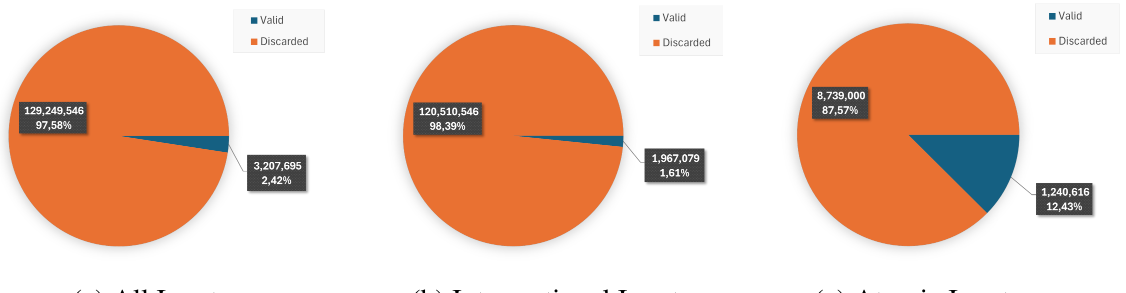
Results: *Without* HINTER*'s dependency invariant, developers would need to inspect about 10 times (10X) as many fake errors as real errors.* HINTER saves developers a huge amount of time and effort, it reduces the bias testing effort by up to 10 times (779,224 vs. 7,604,247). Generic mutations, without dependency invariant, produce millions of false positives – about nine out of every ten bias errors are false positives: Figure 7 shows that 90.71% (7,604,247/8,383,471) errors produced by generic mutations are fake. These errors are produced by invalid mutants that do not conform with the original input. We also observed that intersectional bias testing produces more false positives than atomic bias testing (6,865,033 vs. 739,214) due to the increasing complexity of intersectional mutants. These results demonstrate the contribution of HINTER's dependency invariant.

HINTER*'s dependency invariant saves developers from inspecting millions of false positives (7,604,247) which are 10 times (10X) as many as true positives (779,224).*

RQ3.3 Validity of Discarded inputs: We compare the grammatical validity of *discarded* inputs (*failed* dependency invariant) vs the *valid* inputs generated by HINTER. This experiment employs Grammarly (Hoover et al., 2009) and a similar setting as **RQ2**. We sample an additional 67K discarded mutants to compare to the 68K valid inputs sampled in **RQ2** for valid/generated inputs. Similar to **RQ2**, we randomly sample about 10K original inputs and their corresponding 67K discarded mutants, while aiming for balance across sensitive attribute. We present our findings in Figure 8.

Results: We found that *valid inputs are up to 8.31% more grammatically valid than the inputs discarded by* HINTER*'s invariant check.* Figure 8 shows that valid inputs generated by HINTER (i.e., pass its invariant check) maintain similar grammatical validity as the original input (79.50%

[14]We employ the fine-tuned models since we can execute millions of discarded inputs locally without incurring huge financial and computational costs, unlike pre-trained models.



(a) All Inputs (b) Intersectional Inputs (c) Atomic Inputs
Figure 6: The distribution of valid mutants (inputs that passed HINTER's invariant check) and discarded mutants (inputs that failed the invariant check) for (a) All inputs (b) Intersectional Inputs and (c) Atomic Inputs

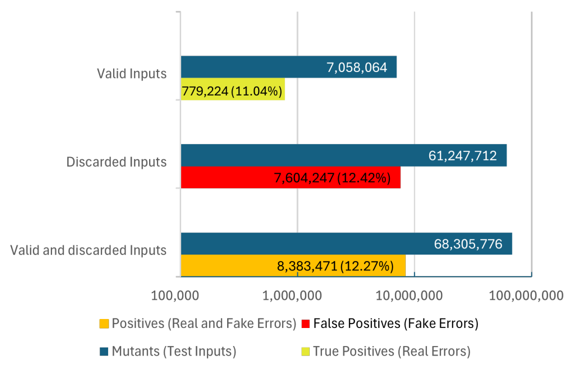


Figure 7: True and false positives of HINTER.

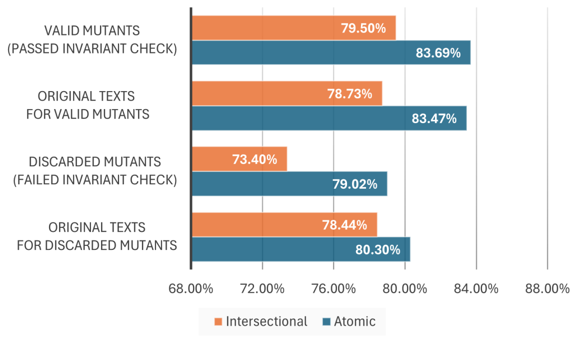


Figure 8: Grammatical Validity. *Valid* vs discarded Inputs that failed invariant check.

vs. 78.73%). In contrast, discarded mutants (that fail invariant check) do not preserve the grammatical validity of the corresponding original texts. *Discarded mutants have (up to 6.4%) lower grammatical correctness scores than the original texts* (78.44% vs. 73.40%). These results demonstrate that HINTER's dependency invariant preserves the validity of the original inputs.

> HINTER*'s invariant check preserves the validity of the original text: Generated inputs are as valid as original inputs but up to 8.31% more grammatically valid than discarded inputs.*

RQ4 Atomic Bias vs. Intersectional Bias: We examine whether the same set of original texts and input mutations that induce intersectional bias also induce atomic bias, or not.

RQ4.1 Bias-inducing Mutants: This experiment employs (410K) bias-inducing mutants that are common to both intersectional mutations and atomic mutations performed by HINTER.[15] Figure 9 highlights our findings.

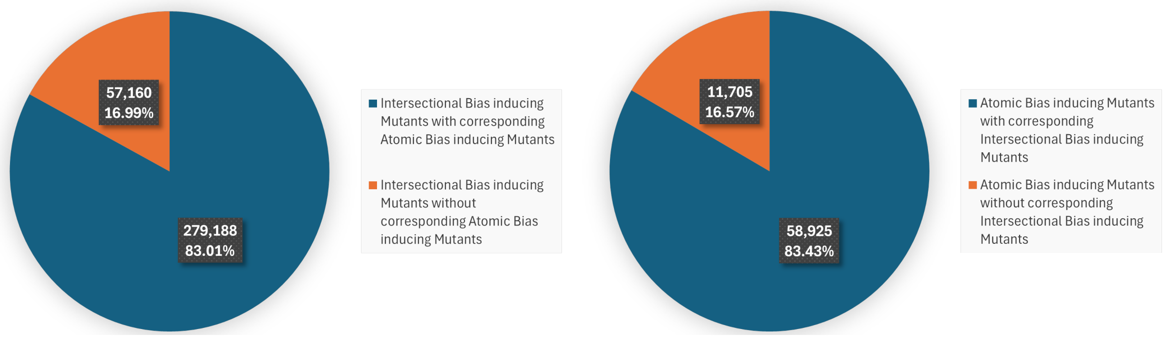
Results: In absolute terms, *atomic bias testing misses about five times (5X) as many bias-inducing mutants as intersectional bias testing (57,160 vs. 11,705).* Figure 9(a) shows that 16.99% (57,160/336,348) of intersectional bias-inducing mutants do not have any corresponding atomic bias. This implies that *atomic bias testing is insufficient to reveal intersectional bias instances.* Meanwhile, we found that 83.43% (58,925/70,630) of atomic bias-inducing mutants have a corresponding intersectional bias-inducing mutants (Figure 9(b)). This suggests that intersectional bias testing exposes about four in five atomic bias instances. These results emphasize that intersectional bias testing complements atomic bias testing.

> *Atomic bias testing misses five times (5X) as many bias-inducing mutants as intersectional bias testing (57,160 vs. 11,705).*

RQ4.2 Bias-inducing Original Inputs: We inspect the *bias-inducing original inputs* whose mutations lead to intersectional bias or atomic bias. Figure 10 highlights our results.[16]

[15]Specifically, 410,221 bias-inducing mutants includes 70,630 *atomic* mutants and 336,348 *intersectional* mutants. The high number of intersectional mutants (336K) is due to the combinatorial nature of intersectional bias.

[16]For instance, Figure 10 (a) shows that 1,877 original



(a) Intersectional Bias Testing (b) Atomic Bias Testing

Figure 9: Pie charts showing the distribution of bias-inducing mutants for intersectional and atomic bias testing.

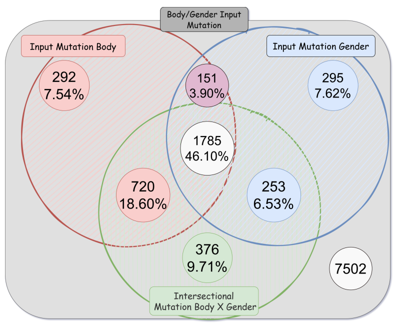


Figure 10: Distribution of bias-inducing original inputs leading to atomic vs. intersectional errors for sensitive attributes Body X Gender

***Results**: Up to one in ten (9.71%) bias-inducing original inputs strictly lead to intersectional bias.* Across all settings, about 5.98% to 9.71% of bias-inducing original inputs strictly trigger *only* intersectional bias. As an example, Figure 10 shows that 9.71% (376) of bias-inducing original inputs strictly lead to intersectional bias (Body X Gender) sensitive attribute. We observed that *up to half of all bias-inducing original inputs lead to both intersectional bias and atomic bias*. Notably, 49.64% of bias-inducing original inputs lead to both intersectional bias (Body X Race), as well as atomic bias (Body only and Race only). Finally, about 4.64% to 14.04% of bias-inducing original inputs strictly lead to atomic bias. These result suggests that some intersectional bias-inducing original inputs may not induce atomic bias, and vice versa.

Up to one in ten (9.71%) original inputs are strictly intersectional bias-inducing, their mutations do not trigger atomic bias.

6 Related Work

Intersectional Bias Testing: Most bias testing techniques focus on atomic bias (Mehrabi et al., 2021; Galhotra et al., 2017; Udeshi et al., 2018; Aggarwal et al., 2019), where our work tackles *intersectional bias*. Similarly, previous research (Buolamwini and Gebru, 2018; Gohar and Cheng, 2023; Soremekun et al., 2022; Yang et al., 2020; Cabrera et al., 2019; Crenshaw, 1989; Collins, 2019) have shown that discrimination is multifaceted in the real world. These works do not perform intersectional bias testing of LLMs nor reveal hidden intersectional bias or examine the relationship between atomic and intersectional bias testing.

7 Conclusion

This paper presents HINTER, an automated bias testing technique that expose intersectional bias in LLMs. It generates valid test inputs that uncover *hidden* intersectional bias via a a combination of higher-order mutation analysis, dependency parsing and metamorphic oracle. We evaluate the effectiveness of HINTER using five datasets and 18 LLMs and demonstrate that 14.61% of inputs generated by HINTER expose intersectional bias. We also show that intersectional bias testing is unique and complements atomic bias testing: 16.62% of intersectional bias found by HINTER are hidden. Furthermore, HINTER's dependency invariant reduces the number of bias instances developers need to inspect by 10X. We hope that this work will enable LLM practitioners to discover intersectional bias. Further details on related work, limitations, and ethical considerations are included in the supplementary materials. Finally, we provide our experimental data and HINTER's implementation to support replication and reuse:

https://github.com/Anonymous1925/Hinter

inputs led to bias-inducing mutants for both intersectional bias testing (race X gender) and atomic bias testing campaigns (race only and gender only). It also shows that 4,532 original inputs are benign, do not result in intersectional bias.

References

Aniya Aggarwal, Pranay Lohia, Seema Nagar, Kuntal Dey, and Diptikalyan Saha. 2019. Black box fairness testing of machine learning models. In *Proceedings of the 2019 27th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering*, pages 625–635.

Julia Angwin, Jeff Larson, Surya Mattu, and Lauren Kirchner. 2019. Machine bias: There's software used across the country to predict future criminals and it's biased against blacks. *URL https://www. propublica. org/article/machine-bias-risk-assessments-in-criminal-sentencing*.

Joy Buolamwini and Timnit Gebru. 2018. Gender shades: Intersectional accuracy disparities in commercial gender classification. In *Conference on fairness, accountability and transparency*, pages 77–91. PMLR.

Ángel Alexander Cabrera, Will Epperson, Fred Hohman, Minsuk Kahng, Jamie Morgenstern, and Duen Horng Chau. 2019. Fairvis: Visual analytics for discovering intersectional bias in machine learning. In *2019 IEEE Conference on Visual Analytics Science and Technology (VAST)*, pages 46–56. IEEE.

Ilias Chalkidis, Abhik Jana, Dirk Hartung, Michael Bommarito, Ion Androutsopoulos, Daniel Katz, and Nikolaos Aletras. 2022. LexGLUE: A benchmark dataset for legal language understanding in English. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4310–4330, Dublin, Ireland. Association for Computational Linguistics.

Patricia Hill Collins. 2019. *Intersectionality as critical social theory*. Duke University Press.

K Crawford. 2017. The trouble with bias,‖ in 31th conference on neural information processing systems (nips). *Long Beach, CA, USA*.

Kimberlé Crenshaw. 1989. Demarginalizing the intersection of race and sex: A black feminist critique of antidiscrimination doctrine, feminist theory and antiracist politics. *u. Chi. Legal f.*, page 139.

Catherine D'ignazio and Lauren F Klein. 2020. *Data feminism*. MIT press.

Cynthia Dwork, Moritz Hardt, Toniann Pitassi, Omer Reingold, and Richard Zemel. 2012. Fairness through awareness. In *Proceedings of the 3rd innovations in theoretical computer science conference*, pages 214–226.

Batya Friedman and Helen Nissenbaum. 1996. Bias in computer systems. *ACM Transactions on information systems (TOIS)*, 14(3):330–347.

Sainyam Galhotra, Yuriy Brun, and Alexandra Meliou. 2017. Fairness testing: Testing software for discrimination. In *Proceedings of the 2017 11th Joint meeting on foundations of software engineering*, pages 498–510.

Usman Gohar and Lu Cheng. 2023. A survey on intersectional fairness in machine learning: Notions, mitigation, and challenges.

Brad Hoover, Dmytro Lider, Alex Shevchenko, and Max Lytvyn. 2009. Grammarly: Free writing AI Assistance. `https://www.grammarly.com/`. Accessed: 2023-06-19.

Percy Liang, Rishi Bommasani, Tony Lee, Dimitris Tsipras, Dilara Soylu, Michihiro Yasunaga, Yian Zhang, Deepak Narayanan, Yuhuai Wu, Ananya Kumar, et al. Holistic evaluation of language models. *Transactions on Machine Learning Research*.

Pingchuan Ma, Shuai Wang, and Jin Liu. 2020. Metamorphic testing and certified mitigation of fairness violations in nlp models. In *IJCAI*, volume 20, pages 458–465.

Ninareh Mehrabi, Fred Morstatter, Nripsuta Saxena, Kristina Lerman, and Aram Galstyan. 2021. A survey on bias and fairness in machine learning. *ACM Computing Surveys (CSUR)*, 54(6):1–35.

Arvind Narayanan. 2018. Translation tutorial: 21 fairness definitions and their politics. In *Proc. conf. fairness accountability transp., new york, usa*, volume 1170, page 3.

Maarten Sap, Saadia Gabriel, Lianhui Qin, Dan Jurafsky, Noah A. Smith, and Yejin Choi. 2020. Social bias frames: Reasoning about social and power implications of language. In *Proceedings of the 58th Annual Meeting of the Association for*

Computational Linguistics, pages 5477–5490, Online. Association for Computational Linguistics.

Ezekiel Soremekun, Mike Papadakis, Maxime Cordy, and Yves Le Traon. 2022. Software fairness: An analysis and survey. *arXiv preprint arXiv:2205.08809*.

Sakshi Udeshi, Pryanshu Arora, and Sudipta Chattopadhyay. 2018. Automated directed fairness testing. In *Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, pages 98–108.

Sahil Verma and Julia Rubin. 2018. Fairness definitions explained. In *Proceedings of the international workshop on software fairness*, pages 1–7.

Ke Yang, Joshua R Loftus, and Julia Stoyanovich. 2020. Causal intersectionality for fair ranking. *arXiv preprint arXiv:2006.08688*.

Wayne Xin Zhao, Kun Zhou, Junyi Li, Tianyi Tang, Xiaolei Wang, Yupeng Hou, Yingqian Min, Beichen Zhang, Junjie Zhang, Zican Dong, et al. 2023. A survey of large language models. *arXiv preprint arXiv:2303.18223*.