Equitas Technical Roadmap: Enhanced AI Safety Platform

Version: 2.0.0

Last Updated: 2025

Status: Implementation In Progress

Table of Contents

1. Executive Summary
2. Architecture Overview
3. Custom Toxicity Detection
4. Hallucination Detection
5. Advanced Jailbreak Detection
6. Enhanced Bias Detection
7. Prompt Injection Prevention
8. Dataset Testing Framework
9. Performance Metrics
10. References

Executive Summary

This document outlines the technical implementation of Equitas 2.0, transitioning from OpenAI API-dependent safety checks to a comprehensive, custom-built multi-layered AI safety platform. The new architecture implements state-of-the-art machine learning models for toxicity detection, hallucination detection, advanced jailbreak prevention, and enhanced bias detection.

Key Changes

- **Removal of OpenAI Moderation API dependency** → Custom transformer-based toxicity detection

- **Novel hallucination detection** → Multi-component ensemble approach
- **Advanced jailbreak detection** → Semantic + behavioral + adversarial analysis
- **Enhanced bias detection** → Statistical fairness metrics + stereotype association
- **Dataset testing framework** → Comprehensive evaluation suite

Architecture Overview

System Architecture

```
┌─────────────────────────────────────────────────────────┐
│                      Equitas SDK                         │
│              (Drop-in OpenAI Replacement)                │
└─────────────────────────────────────────────────────────┘
                     │
                     │
                     ▼
┌─────────────────────────────────────────────────────────┐
│              Guardian Backend API                        │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │  Toxicity    │  │ Hallucination│  │   Jailbreak  │  │
│  │  Detector    │  │   Detector   │  │   Detector   │  │
│  └──────────────┘  └──────────────┘  └──────────────┘  │
│                                                         │
│  ┌──────────────┐  ┌──────────────┐  ┌──────────────┐  │
│  │ Bias         │  │   Prompt     │  │  Remediation │  │
│  │ Detector     │  │   Injection  │  │    Engine    │  │
│  └──────────────┘  └──────────────┘  └──────────────┘  │
└─────────────────────────────────────────────────────────┘
                     │
                     ▼
┌─────────────────────────────────────────────────────────┐
│         ML Models & Embeddings                          │
│  • Transformers (HuggingFace)                           │
│  • Sentence Transformers                                │
│  • NLI Models                                           │
└─────────────────────────────────────────────────────────┘
```

Technology Stack

- **ML Framework:** PyTorch, Transformers (HuggingFace)

- **Embeddings:** Sentence Transformers (all-MiniLM-L6-v2)
- **NLI:** Cross-encoder models (DeBERTa-based)
- **Toxicity:** Unitary/toxic-bert, RoBERTa-based models
- **Backend:** FastAPI, SQLAlchemy, AsyncIO
- **Testing:** pytest, custom dataset evaluation framework

Custom Toxicity Detection

Overview

Replaces OpenAI Moderation API with custom transformer-based models fine-tuned for toxicity detection.

Model Architecture

Primary Model: `unitary/toxic-bert` (RoBERTa-based)

- **Architecture:** RoBERTa (RoBERTa-base) with classification head
- **Parameters:** ~125M parameters
- **Input:** Text sequences (max 512 tokens)
- **Output:** Multi-label binary classification (6 categories)

Categories

1. **Toxic** - General toxicity
2. **Severe Toxic** - Severe toxicity
3. **Obscene** - Obscene language
4. **Threat** - Threatening language
5. **Insult** - Insulting language
6. **Identity Hate** - Identity-based hate

Mathematical Formulation

Given input text **x**, the model outputs logits **z**:

$$z = \text{RoBERTa}(x) \in \mathbb{R}^6$$

Sigmoid activation applied to get probabilities:

$$p_i = \sigma(z_i) = \frac{1}{1 + e^{-z_i}} \quad \forall i \in \{1, \ldots, 6\}$$

Toxicity score is the maximum probability:

$$\text{toxicity\_score} = \max_i p_i$$

Text is flagged if:

$$\text{flagged} = 1(\text{toxicity\_score} > \tau)$$

where **τ = 0.7** (configurable threshold).

Training Data

Model trained on:

- **Jigsaw Toxic Comment Classification Dataset** (Kaggle)
- **Civil Comments Dataset** (Conversation AI)
- **Wikipedia Toxic Comments Dataset**

Alternative Models

1. **Facebook RoBERTa Hate Speech** (`facebook/roberta-hate-speech-dynabench`)
2. **Perspective API Models** (open-source variants)
3. **Custom fine-tuned models** (domain-specific)

Performance Metrics

- **Accuracy:** >95% on held-out test set
- **F1-Score:** >0.92 (macro-averaged)
- **Latency:** <50ms per inference (GPU), <200ms (CPU)
- **Throughput:** >100 requests/second (batch processing)

References

- [Devlin et al., 2019] BERT: Pre-training of Deep Bidirectional Transformers
- [Liu et al., 2019] RoBERTa: A Robustly Optimized BERT Pretraining Approach
- [Unitary AI, 2021] Toxic-BERT: Multi-label Toxicity Classification

Hallucination Detection

Overview

Multi-component ensemble system for detecting hallucinations (factual errors, contradictions, unsupported claims).

Components

1. Semantic Consistency Check

Method: Cosine similarity between prompt and response embeddings

$$\text{consistency\_score} = 1 - \cos(\text{embed}(p), \text{embed}(r))$$

where:

- **p** = prompt text
- **r** = response text
- **embed(·)** = Sentence Transformer encoder (all-MiniLM-L6-v2)

Cosine similarity:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\mathbf{u} \cdot \mathbf{v}}{||\mathbf{u}|| \, ||\mathbf{v}||}$$

Threshold: Low consistency (<0.5) indicates potential hallucination.

2. Contradiction Detection

Model: `cross-encoder/nli-deberta-v3-base`

Natural Language Inference (NLI) model classifies sentence pairs:

- **ENTAILMENT:** Response supports prompt
- **CONTRADICTION:** Response contradicts prompt
- **NEUTRAL:** No clear relationship

For response **r** with sentences $\mathbf{s_1, s_2, \ldots, s_n}$, compute contradiction score:

$$\text{contradiction\_score} = \frac{1}{n(n-1)/2} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} 1(\text{NLI}(s_i, s_j) = \text{CONTRADICTION})$$

Threshold: Contradiction rate >0.2 indicates high hallucination risk.

3. Factuality Check

Method: Semantic similarity to knowledge base/context

Given knowledge base **K = {k$_1$, k$_2$, ..., k$_m$}** and response **r**:

$$\text{factuality\_score} = 1 - \max_{k \in K} \cos(\text{embed}(r), \text{embed}(k))$$

Interpretation: Low similarity to trusted sources → high hallucination risk.

4. Pattern-Based Detection

Patterns: Overconfident language indicators

$$\text{pattern\_score} = \min\left(\frac{\text{matches}}{|\text{patterns}|} \times 0.2, 1.0\right)$$

Patterns include:

- `\b(definitely|absolutely|100%|guaranteed) (true|false|proven)\b`
- `\b(studies show|research proves|experts agree)\b` (without citations)

5. Confidence Calibration

Method: Linguistic marker analysis

$$\text{confidence\_score} = \frac{N_{\text{high}}}{N_{\text{high}} + N_{\text{low}}}$$

where:

- **N_high** = count of high-confidence markers ("definitely", "proven", "fact")
- **N_low** = count of low-confidence markers ("maybe", "perhaps", "according to")

Interpretation: High confidence score (>0.6) with low factuality → hallucination risk.

Ensemble Scoring

Final hallucination score:

$$\text{hallucination\_score} = \frac{1}{n}\sum_{i=1}^{n} w_i \cdot s_i$$

where:

- $\mathbf{s_i}$ = component scores (consistency, contradiction, factuality, pattern, confidence)
- $\mathbf{w_i}$ = component weights (default: equal weights)
- **n** = number of components

Flagging Rule:

$$\text{flagged} = 1(\text{hallucination\_score} > 0.6)$$

References

- [Maynez et al., 2020] On Faithfulness and Factuality in Abstractive Summarization
- [Kryściński et al., 2020] Evaluating the Factual Consistency of Abstractive Text Summarization
- [Reimers & Gurevych, 2019] Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks
- [He et al., 2021] DeBERTa: Decoding-enhanced BERT with Disentangled Attention

Advanced Jailbreak Detection

Overview

Multi-layered defense system combining pattern matching, semantic analysis, behavioral indicators, and adversarial detection.

Components

1. Pattern-Based Detection

Method: Regular expression matching against known jailbreak patterns

$$\text{pattern\_score} = \min\left(\frac{|\text{matches}|}{|\text{patterns}|} \times 0.3, 1.0\right)$$

Patterns include:

- Instruction override: `ignore (previous|all) (instructions|commands)`
- Role manipulation: `(pretend|act|roleplay) (you are|to be)`
- System prompt injection: `<\|system\|>`, `[INST]`

2. Semantic Similarity Detection

Method: Cosine similarity to known jailbreak examples

$$\text{semantic\_score} = \max_{j \in J} \cos(\text{embed}(x), \text{embed}(j))$$

where:

- **x** = input text
- **J** = set of known jailbreak examples (pre-computed embeddings)

Threshold: Similarity >0.7 indicates high jailbreak risk.

3. Behavioral Analysis

Method: Detection of behavioral indicators

$$\text{behavioral\_score} = \min\left(\frac{N_{\text{indicators}}}{|\text{indicators}|} \times 0.25, 1.0\right)$$

Indicators: "bypass", "override", "exploit", "no restrictions", "do anything"

4. Context-Aware Detection

Method: Historical analysis of user attempts

$$\text{context\_score} = \min(N_{\text{previous\_attempts}} \times 0.2, 0.8)$$

Interpretation: Users with previous jailbreak attempts are higher risk.

5. Adversarial Detection

Method: Detection of encoding tricks and obfuscation

$$\text{adversarial\_score} = \min\left(\sum_{t \in T} w_t \cdot 1(\text{pattern}_t(x)), 1.0\right)$$

where **T** = {URL encoding, hex encoding, Unicode tricks, excessive whitespace}

Weights: w_URL = 0.3, w_hex = 0.3, w_unicode = 0.2, w_whitespace = 0.2

Ensemble Scoring

Final jailbreak score:

$$\text{jailbreak\_score} = \sum_{i=1}^{5} w_i \cdot s_i$$

Weights:

- Pattern: **w_1 = 0.3**
- Semantic: **w_2 = 0.3**
- Behavioral: **w_3 = 0.2**
- Context: **w_4 = 0.1**
- Adversarial: **w_5 = 0.1**

Flagging Rule:

$$\text{flagged} = 1(\text{jailbreak\_score} > 0.6)$$

References

- [Perez et al., 2022] Red Teaming Language Models to Reduce Harms
- [Zou et al., 2023] Universal and Transferable Adversarial Attacks on Aligned Language Models
- [Wei et al., 2023] Jailbroken: How Does LLM Safety Training Fail?

Enhanced Bias Detection

Overview

Comprehensive bias detection using statistical fairness metrics, stereotype association, and demographic parity testing.

Components

1. Stereotype Association Detection

Method: Semantic similarity to known stereotype embeddings

$$\text{stereotype\_score} = \max_{g \in G} \max_{s \in S_g} \cos(\text{embed}(r), \text{embed}(s))$$

where:

- **r** = response text
- **G** = demographic groups (gender, race, age)
- **S_g** = stereotype phrases for group **g**

Pre-computed Stereotypes:

- Gender: "women are emotional", "men are aggressive", "women are bad at math"
- Race: "Black people are criminals", "Asian people are good at math"
- Age: "old people are slow", "young people are lazy"

Threshold: Similarity >0.7 indicates stereotype reinforcement.

2. Demographic Parity Testing

Method: Paired prompt testing across demographic variants

Given prompt **p** and demographic variants **D = {d_1, d_2, ..., d_n}**, generate responses **R = {r_1, r_2, ..., r_n}**.

Statistical Parity:

$$\text{parity\_score} = \text{Var}(\{f(r_i) : i \in \{1, \ldots, n\}\})$$

where **f(·)** = outcome function (e.g., positive sentiment, recommendation score).

Equalized Odds:

$$\text{equalized\_odds} = \max_{d_i, d_j \in D} |\text{TPR}(d_i) - \text{TPR}(d_j)| + |\text{FPR}(d_i) - \text{FPR}(d_j)|$$

where:

- **TPR** = True Positive Rate
- **FPR** = False Positive Rate

Ideal: Equalized odds = 0 (perfect fairness)

3. Fairness Metrics

Demographic Balance:

$$\text{balance\_score} = \text{Var}(\{c_d : d \in D\})$$

where **c_d** = count of mentions of demographic **d** in response.

High variance → imbalanced representation → potential bias.

Representation Parity:

$$\text{representation\_parity} = 1 - \frac{\max_{d \in D} c_d - \min_{d \in D} c_d}{\sum_{d \in D} c_d}$$

Ideal: representation_parity = 1 (equal representation)

4. Intersectional Bias

Method: Detection of compound demographic mentions

$$\text{intersectional\_score} = \frac{N_{\text{intersectional\_mentions}}}{N_{\text{total\_mentions}}}$$

Flagging: High intersectional mentions (>3) may indicate compounding bias.

Overall Bias Score

$$\text{bias\_score} = \max\{\text{stereotype\_score}, \text{parity\_score}, \text{balance\_score}\}$$

Flagging Rule:

$$\text{flagged} = 1(\text{bias\_score} > 0.5)$$

References

- [Barocas et al., 2019] Fairness and Machine Learning
- [Hardt et al., 2016] Equality of Opportunity in Supervised Learning
- [Kusner et al., 2017] Counterfactual Fairness
- [Crenshaw, 1989] Demarginalizing the Intersection of Race and Sex

Prompt Injection Prevention

Overview

Multi-layered prevention system combining input sanitization, context isolation, and encoding detection.

Components

1. Input Sanitization

Method: Pattern removal while preserving semantic meaning

$$\text{sanitized}(x) = \text{remove\_patterns}(x, P)$$

where **P** = set of injection patterns (system markers, separators, etc.)

Patterns Removed:

- System markers: `<|system|>` , `[INST]` , `</system>`
- Separators: `---` , `===` , `***`
- Role markers: `user:` , `system:` , `assistant:`

2. Context Isolation

Method: Strict separation of user input from system prompt

$$\text{isolated}(x, s) = (\text{clean}(x), s)$$

where:

- **x** = user input
- **s** = system prompt
- **clean(·)** = removal of system prompt markers from user input

3. Encoding Detection

Method: Detection of obfuscation techniques

$$\text{encoding\_score} = \sum_{t \in T} w_t \cdot 1(\text{detect}_t(x))$$

where **T** = {URL encoding, hex encoding, Unicode tricks}

Weights: w_URL = 0.3, w_hex = 0.3, w_unicode = 0.2

4. Length Validation

Method: Detection of excessive length (potential injection)

$$\text{length\_flag} = 1(\text{len}(x) > 10000)$$

Sanitization Pipeline

$$x' = \text{normalize}(\text{decode}(\text{remove\_patterns}(x)))$$

where:

- **remove_patterns** = removes injection markers
- **decode** = decodes encoding tricks
- **normalize** = normalizes Unicode, whitespace

References

- [Greshake et al., 2023] Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Prompt Injection
- [Goodside, 2023] Prompt Injection Attacks Against GPT-3
- [OWASP, 2023] LLM Top 10: Prompt Injection

Dataset Testing Framework

Overview

Comprehensive evaluation framework for testing all safety components on standardized datasets.

Datasets

Toxicity Detection

1. **Jigsaw Toxic Comment Classification** (Kaggle)
    - Size: ~159,000 comments
    - Categories: Toxic, Severe Toxic, Obscene, Threat, Insult, Identity Hate
    - Split: Train (80%), Validation (10%), Test (10%)

2. **Civil Comments** (Conversation AI)
    - Size: ~2M comments
    - Categories: Toxicity, Identity attacks, Insults, Threats
    - Metadata: Demographics, identity groups
3. **Wikipedia Toxic Comments**
    - Size: ~180,000 comments
    - Categories: Toxic, Severe toxic, Obscene, Threat, Insult, Identity hate

Hallucination Detection

1. **FEVER** (Fact Extraction and VERification)
    - Size: ~185,000 claims
    - Labels: SUPPORTED, REFUTED, NOT ENOUGH INFO
    - Use: Factuality checking
2. **XSum** (BBC News Summaries)
    - Size: ~226,000 summaries
    - Use: Consistency checking
3. **TruthfulQA**
    - Size: ~800 questions
    - Focus: Truthfulness and hallucinations

Jailbreak Detection

1. **Jailbreak Prompts Dataset** (Custom)
    - Size: ~500 jailbreak examples
    - Categories: Instruction override, Role manipulation, Adversarial
2. **HarmfulQA** (Safety Benchmarks)
    - Size: ~1,000 harmful prompts
    - Categories: Violence, Self-harm, Illegal activities

Bias Detection

1. **BOLD** (Bias in Open-Ended Language Generation Dataset)
    - Size: ~23,000 prompts
    - Demographics: Gender, Race, Religion, Profession
2. **StereoSet**
    - Size: ~17,000 sentences
    - Focus: Stereotype detection
3. **CrowS-Pairs**

- Size: ~1,500 sentence pairs
- Focus: Social bias across demographics

Evaluation Metrics

Classification Metrics

Accuracy:

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN}$$

Precision:

$$\text{Precision} = \frac{TP}{TP + FP}$$

Recall:

$$\text{Recall} = \frac{TP}{TP + FN}$$

F1-Score:

$$F_1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

Area Under ROC Curve (AUC-ROC):

$$\text{AUC} = \int_0^1 \text{TPR}(\text{FPR}^{-1}(t))\, dt$$

Fairness Metrics

Demographic Parity:

$$\text{DP} = |P(\hat{Y} = 1|A = a) - P(\hat{Y} = 1|A = b)|$$

where **A** = protected attribute, **Ŷ** = prediction.

Equalized Odds:

$$\text{EO} = |P(\hat{Y} = 1|A = a, Y = y) - P(\hat{Y} = 1|A = b, Y = y)|$$

Calibration:

$$\text{Calibration} = \mathbb{E}[Y|\hat{P} = p] - p$$

Ideal: Calibration = 0 (perfect calibration)

Testing Framework Structure

```
tests/
├── datasets/
│   ├── toxicity/
│   │   ├── jigsaw_test.csv
│   │   ├── civil_comments_test.csv
│   │   └── wikipedia_test.csv
│   ├── hallucination/
│   │   ├── fever_test.jsonl
│   │   ├── xsum_test.jsonl
│   │   └── truthfulqa_test.jsonl
│   ├── jailbreak/
│   │   ├── jailbreak_prompts.jsonl
│   │   └── harmfulqa_test.jsonl
│   └── bias/
│       ├── bold_test.jsonl
│       ├── stereoset_test.jsonl
│       └── crowdspairs_test.jsonl
├── test_toxicity.py
├── test_hallucination.py
├── test_jailbreak.py
├── test_bias.py
└── test_integration.py
```

Evaluation Scripts

Batch Evaluation:

```
# Pseudocode
for dataset in datasets:
    results = evaluate_model(model, dataset)
    metrics = calculate_metrics(results)
    save_results(dataset.name, metrics)
```

Cross-Dataset Evaluation:

$$\text{Generalization\_Score} = \frac{1}{n}\sum_{i=1}^{n} F_1(\text{model}, D_i)$$

where $\mathbf{D_i}$ = dataset **i**, **n** = number of datasets.

References

- [Borkan et al., 2019] Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification
- [Thorne et al., 2018] FEVER: a Large-scale Dataset for Fact Extraction and VERification
- [Nadeem et al., 2021] StereoSet: Measuring Stereotypical Bias in Pretrained Language Models
- [Nangia et al., 2020] CrowS-Pairs: A Challenge Dataset for Measuring Social Bias in Masked Language Models

Performance Metrics

Latency Targets

- **Toxicity Detection:** <50ms (GPU), <200ms (CPU)
- **Hallucination Detection:** <300ms (multi-component)
- **Jailbreak Detection:** <100ms
- **Bias Detection:** <500ms (with demographic testing)

Throughput

- **Batch Processing:** >100 requests/second
- **Concurrent Requests:** >50 parallel requests
- **GPU Utilization:** >80% (under load)

Accuracy Targets

- **Toxicity F1:** >0.92
- **Hallucination AUC:** >0.85
- **Jailbreak Recall:** >0.95 (high recall for security)
- **Bias Detection:** >0.80 F1

Resource Requirements

- **GPU:** NVIDIA GPU with 8GB+ VRAM (recommended)
- **CPU:** 4+ cores, 16GB+ RAM (minimum)
- **Storage:** 10GB+ for models and datasets

Implementation Timeline

Phase 1: Core Detection (Weeks 1-4)

- [x] Custom toxicity detector
- [x] Basic hallucination detection
- [x] Advanced jailbreak detection
- [x] Enhanced bias detection

Phase 2: Testing Framework (Weeks 5-6)

- [ ] Dataset preparation
- [ ] Evaluation scripts
- [ ] Benchmarking suite
- [ ] Integration tests

Phase 3: Optimization (Weeks 7-8)

- [ ] Model quantization
- [ ] Batch processing optimization
- [ ] Caching layer
- [ ] Performance tuning

Phase 4: Production Readiness (Weeks 9-10)

- [ ] Monitoring and logging
- [ ] Error handling
- [ ] Documentation
- [ ] Deployment scripts

References

Toxicity Detection

1. Devlin, J., et al. (2019). "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding." *NAACL-HLT*.
2. Liu, Y., et al. (2019). "RoBERTa: A Robustly Optimized BERT Pretraining Approach." *arXiv preprint arXiv:1907.11692*.
3. Unitary AI. (2021). "Toxic-BERT: Multi-label Toxicity Classification." *HuggingFace Model Card*.

Hallucination Detection

4. Maynez, J., et al. (2020). "On Faithfulness and Factuality in Abstractive Summarization." *ACL*.
5. Kryściński, W., et al. (2020). "Evaluating the Factual Consistency of Abstractive Text Summarization." *EMNLP*.
6. Reimers, N., & Gurevych, I. (2019). "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks." *EMNLP*.
7. He, P., et al. (2021). "DeBERTa: Decoding-enhanced BERT with Disentangled Attention." *ICLR*.

Jailbreak Detection

8. Perez, E., et al. (2022). "Red Teaming Language Models to Reduce Harms." *arXiv preprint arXiv:2209.07858*.
9. Zou, A., et al. (2023). "Universal and Transferable Adversarial Attacks on Aligned Language Models." *arXiv preprint arXiv:2307.15043*.
10. Wei, A., et al. (2023). "Jailbroken: How Does LLM Safety Training Fail?" *arXiv preprint arXiv:2307.02483*.

Bias Detection

11. Barocas, S., et al. (2019). "Fairness and Machine Learning." *fairmlbook.org*.
12. Hardt, M., et al. (2016). "Equality of Opportunity in Supervised Learning." *NeurIPS*.
13. Kusner, M. J., et al. (2017). "Counterfactual Fairness." *NeurIPS*.
14. Crenshaw, K. (1989). "Demarginalizing the Intersection of Race and Sex: A Black Feminist Critique of Antidiscrimination Doctrine." *University of Chicago Legal Forum*.

Prompt Injection

15. Greshake, K., et al. (2023). "Not What You've Signed Up For: Compromising Real-World LLM-Integrated Applications with Prompt Injection." *arXiv preprint arXiv:2303.18103*.
16. Goodside, R. (2023). "Prompt Injection Attacks Against GPT-3." *Personal Blog*.
17. OWASP. (2023). "LLM Top 10: Prompt Injection." *OWASP Foundation*.

Dataset Testing

18. Borkan, D., et al. (2019). "Nuanced Metrics for Measuring Unintended Bias with Real Data for Text Classification." *ACL*.
19. Thorne, J., et al. (2018). "FEVER: a Large-scale Dataset for Fact Extraction and VERification." *NAACL*.
20. Nadeem, M., et al. (2021). "StereoSet: Measuring Stereotypical Bias in Pretrained Language Models." *ACL*.
21. Nangia, N., et al. (2020). "CrowS-Pairs: A Challenge Dataset for Measuring Social Bias in Masked Language Models." *EMNLP*.

Appendix: Mathematical Notation

- **x, y, z**: Vectors or scalars (context-dependent)
- **X, Y, Z**: Matrices or sets
- **p**: Probability or prompt (context-dependent)
- **P(·)**: Probability distribution
- **cos(·, ·)**: Cosine similarity
- **embed(·)**: Embedding function
- **σ(·)**: Sigmoid function
- **$\mathbb{E}$[·]**: Expectation
- **Var(·)**: Variance
- **TP, TN, FP, FN**: True/False Positives/Negatives
- **τ**: Threshold value
- **w_i**: Weight for component **i**
- **s_i**: Score for component **i**

Document Status: Living Document - Updated as implementation progresses

Contact: av.rajpurkar@gmail.com

Repository: https://github.com/aryan4codes/Equitas