

**Natural Language Processing (SE-345)**

**NLP-Project-Module-1+2+3**

Instructor: **Dr. Kanwal Yousaf**

Student:MuhammadAmad Ahmad(22-SE-**10**)

Project: **FairWrite– Bias detection in text**

Date: **May 16, 2025**

📑**Table of Contents**

[🎯Project Goal 3](#_heading=h.lrsudli0gves)

[🧭Introduction 3](#_heading=h.mfum785mfp7h)

[Broad Context and Motivation 3](#_heading=h.eal1h8mu71j5)

[Research Gap 4](#_heading=h.617cr98df3c2)

[Problem Statement 4](#_heading=h.rd6js22sf15m)

[Objectives, Scope, and Contribution 4](#_heading=h.1blit4opm45n)

[📚Literature Review 5](#_heading=h.2qay06bp6k3)

[Bias in AI and NLP 5](#_heading=h.jgksqta6pfgm)

[Bias Mitigation Techniques 5](#_heading=h.ugcyrkqx2ar0)

[Pre-Processing Techniques 5](#_heading=h.vaciov5qx9yo)

[In-Processing Techniques 5](#_heading=h.qvcu07h9640j)

[Post-Processing Techniques 6](#_heading=h.q1v8kl75hw1)

[Existing Bias Detection and Mitigation Frameworks 6](#_heading=h.8hrtycn6koab)

[🧪Methodology 7](#_heading=h.vufqm0518t9o)

[Vector Embedding Implementation 7](#_heading=h.cn0jz1629510)

[Data Collection 7](#_heading=h.ik44td2vzuke)

[Data Preprocessing 7](#_heading=h.6ft3nwpmvfe7)

[🧠Embeddings using BERT 8](#_heading=h.ivpv7rgce6dn)

[🧠Embeddings using GloVe 9](#_heading=h.sf468dnnb40k)

[⚙️Applying Neural Networks 10](#_heading=h.oh1gq0lef38g)

[Bert Embeddings >> Simple NN (binary classification) 10](#_heading=h.v7y12wcac7gr)

[BERT Embeddings >> Transformer Based Model (multiclass classification) 12](#_heading=h.d5p3ediz25gt)

[BERT Embeddings >> BiLSTM model 14](#_heading=h.wwco5chipgyq)

[🤖LLM Implementation 17](#_heading=h.x2r2nb6swunj)

[LLM Evaluation Results (Gemini 2.0 Flash) 17](#_heading=h.bbw1egy2xhg2)

[📈Results 19](#_heading=h.vv6lpv4iqs4c)

[1. Binary Classification (Biased, Unbiased) 19](#_heading=h.1wb3lrkhau7z)

[📊Binary Classification Report 20](#_heading=h.vydvkywky0sw)

[2. Multiclass Classification (Biased, Unbiased, Opinion) 21](#_heading=h.y7252sj4w9g9)

[a) BiLSTM Model 21](#_heading=h.e2iugshtokdf)

[📊Classification Report (BiLSTM): 21](#_heading=h.tsnzdfz51j0h)

[b) Transformer-Inspired Dense Model 21](#_heading=h.q1upawyp73b)

[📊Classification Report (Transformer Model): 23](#_heading=h.hfpvbwi3vt8y)

[🧩Comparative Analysis 23](#_heading=h.1ic7ir9sq8p3)

[📖References 26](#_heading=h.a0nrmjulujm)

# 🎯Project Goal

The goal of this project is to develop a specialized **AI**-**driven** **bias** **detection** and **mitigation** system that **enhances** **fairness** in textual content. The system will identify biased words and phrases, classify text based on bias presence, and **suggest** **neutral** **alternatives**. While existing solutions like **Dbias** provide a research framework, our approach aims to fine-tune and extend bias detection beyond news articles to general text, ensuring applicability in broader domains such as blogs, opinion pieces, and everyday writing. This research will contribute to the field of **NLP** by demonstrating how AI can be utilized for ethical and unbiased text processing.

# 🧭Introduction

## Broad Context and Motivation

Bias in textual content, whether in news articles, blogs, or everyday writing, can reinforce **harmful stereotypes**, **misinform** the **public**, and lead to **unfair** **decision**-**making**. With the increasing reliance on AI-driven content generation and analysis, ensuring fairness in text has become crucial. While large language models (LLMs) like GPT-4, GPT-3, llama3.4, phi4, mistral, gemma, and deepseek-r1 aim for neutrality, they still reflect biases present in their training data. There is a growing need for specialized AI models that not only detect but actively mitigate biases in text, ensuring fairness and inclusiveness in various forms of written communication.

## Research Gap

While Dbias provides a promising framework for bias detection and mitigation, it remains primarily a **research** **tool** rather than a widely accessible application for public use. Original paper says, “*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.*” Most existing llms are trained on vast datasets that may contain biases and thus are susceptible to generating **biased** **content**. Furthermore, there is a lack of domain-adaptive bias mitigation tools that cater to a broader audience beyond journalistic content, such as bloggers, academic writers, and **general** **users**. A practical, web-based, or API-driven solution that integrates bias detection and debiasing into daily writing tools is still missing.

## Problem Statement

This research aims to develop an AI-driven bias detection and mitigation system that is accessible beyond the research community. Unlike generic LLMs that generate text with potential biases, our model will focus specifically on identifying biased words and phrases, masking them, and suggesting neutral alternatives. By leveraging transformer-based models and fine-tuning on diverse textual data beyond news articles, we seek to create an adaptive system that ensures fairness in different forms of writing while maintaining semantic integrity.

## Objectives, Scope, and Contribution

The objective of this research is to design and implement a web-based or API-integrated bias mitigation tool that **extends** the capabilities of existing frameworks like Dbias. This system will:

1. Detect biases in a variety of text formats.
2. Recognize and highlight biased phrases.
3. Provide unbiased alternative wordings.
4. Integrate into commonly used writing platforms. (not intended for semester project)

The scope of this work extends beyond news articles to general writing, ensuring that fairness principles apply to blogs, opinion pieces, and everyday user-generated content. Our contribution lies in making bias mitigation tools more practical, accessible, and adaptable to real-world applications.

# 📚Literature Review

## Bias in AI and NLP

With the increasing reliance on AI in sensitive domains such as journalism, social media, hiring, and legal decision-making, concerns over **bias and fairness in NLP models** have gained significant attention. Studies have shown that machine learning models trained on large text corpora often **inherit societal biases**, leading to discriminatory outputs. For example, the COMPAS algorithm exhibited racial biases by incorrectly predicting a higher likelihood of recidivism for African American defendants. Similarly, biased AI systems have been observed in hiring algorithms, targeted advertising, and medical recommendations. These issues highlight the **need for bias detection and mitigation techniques** to ensure fairness in AI-driven decision-making.

Bias in NLP is often categorized into **individual fairness** (ensuring similar individuals receive similar predictions) and **group fairness** (ensuring unbiased outcomes across demographic groups). However, achieving fairness in NLP remains challenging due to the complexity of language and the subjective nature of bias. Therefore, researchers have proposed various algorithmic techniques to **detect, analyze, and mitigate bias** at various stages of the NLP pipeline.

## Bias Mitigation Techniques

Bias mitigation in NLP models is typically approached using **pre-processing, in-processing, and post-processing methods**.

### Pre-Processing Techniques

Pre-processing techniques focus on **modifying the training data** to reduce bias before model training. **Reweighing** adjusts the weights of training samples based on their group attributes to ensure fairness across different demographics. The **Learning Fair Representations** (LFR) method encodes data into a transformed representation that minimizes the effect of protected attributes such as gender or race. Similarly, the **Disparate Impact Remover** modifies feature values in a dataset while preserving rank ordering to improve group fairness. These methods help **reduce bias at the data level**, but they do not address biases introduced during model training or prediction.

### In-Processing Techniques

In-processing techniques **adjust the learning process** of machine learning models to enforce fairness constraints. **Prejudice Remover** applies a fairness-aware regularization term during training to reduce discrimination in model outputs. **Adversarial De-biasing** employs an adversarial framework where a secondary model attempts to predict a sensitive attribute (e.g., gender, race) while the primary model learns to minimize this prediction, thereby ensuring fairness. Another notable approach is **Exponentiated Gradient Reduction**, which iteratively adjusts classification decisions to optimize fairness metrics while minimizing performance loss. These techniques improve fairness **within the model**, but they require modification of model architectures, making them less accessible for pre-trained NLP models.

### Post-Processing Techniques

Post-processing methods adjust model predictions **after inference** to improve fairness. **Equalized Odds Post-Processing** modifies the output labels of a classifier using linear programming to equalize false positive and false negative rates across demographic groups. **Calibrated Equalized Odds** builds on this by optimizing score outputs to ensure unbiased decision thresholds. **Reject Option Classification** selectively modifies predictions in uncertain regions to favor disadvantaged groups. These methods are useful when retraining models is not feasible, but they do not prevent biases from being learned during training.

## Existing Bias Detection and Mitigation Frameworks

A number of toolkits have been developed to evaluate and mitigate bias in AI models. **AI Fairness 360 (AIF360)** is an open-source library that provides fairness metrics and bias mitigation algorithms for different ML pipelines. **FairML** uses influence functions to quantify the contribution of input features to model predictions, helping researchers detect biased decision-making patterns. **FairTest** applies statistical testing to discover biases in machine learning models based on protected attributes. These toolkits offer valuable resources for researchers but are often designed **for technical users** rather than the public.

Among domain-specific solutions, **Dbias** is a recent framework developed to mitigate biases in **news articles** using a pipeline of **three Transformer-based models**:

(1) a **DistilBERT classifier** for bias detection.

(2) a **RoBERTa-based Named Entity Recognition (NER) model** for bias recognition.

(3) a **Masked Language Model (MLM)** for bias mitigation.

While Dbias provides an effective **research-oriented solution**, it remains a framework rather than a **widely accessible application** for everyday writers. This highlights the need for a more **user-friendly system** that integrates bias detection and mitigation into general writing workflows, including blog posts, academic writing, and personal communication.

# 🧪Methodology

## Vector Embedding Implementation

### Data Collection

We used **MBIC – A Media Bias Annotation Dataset.** MBIC is the first available dataset about media bias reporting detailed information on annotator characteristics and their individual background. The first sample of statements represents various media bias instances. We are following the research paper on Dbias for comparative analysis, so we used the single dataset.

#### Data Example:

Sentence: “*The increasingly bitter dispute between American women’s national soccer team and the U.S. Soccer Federation spilled onto the field Wednesday night when players wore their warm-up jerseys inside outing a protest before their 3-1 victory over Japan.”*

Outlet: MSNBC

Topic: sport

Type: left

Label\_Bias: Non-biased

Label\_Opinion: Entirely factual

Biased\_Words: ['bitter']

### Data Preprocessing

Took the important columns for embeddings and model training ['sentence', 'topic', 'type', 'biased\_words4', 'Label\_bias', 'Label\_opinion'], Dropped the under representative label “No Agreement”, and then worked the problem as both:

1. Binary Classification
2. Multiclass Classification

Although the original paper mentioned the work with two class Biased and Unbiased. But we also did create a third label “Opinion” based on the following case:

If Label\_bias == ‘Biased’ and Label\_opinion == ‘Writer’s Opinion’ we label the final label as “Opinion”, this balances the three classes perfectly.

For both the sentence data was lowercased and ridden of any punctuation. We One Hot Encoded the 2 labels in case (a) and 3 labels in case (b). Also, we encoded the 'topic' and 'type'

columns.

The data was split into train and validation sets based on class equalization i.e. Using stratified split.

### 🧠Embeddings using BERT

tokenizer = BertTokenizer.from\_pretrained('bert-base-uncased')

model\_bert = BertModel.from\_pretrained('bert-base-uncased')

def get\_bert\_embeddings(text):

    try:

        inputs = tokenizer(text, return\_tensors="pt", truncation=True, padding=True, max\_length=512)

        with torch.no\_grad():

            outputs = model\_bert(\*\*inputs)

        return outputs.last\_hidden\_state.mean(dim=1).squeeze().numpy()

    except Exception as e:

        print(f"Error processing: {text} → {e}")

        return np.zeros(768)

##### Tokenization

1. Input text is first normalized (lowercased for 'uncased' models)
2. The tokenizer splits text into WordPiece tokens (subword units)
3. Special tokens are added: [CLS] at beginning and [SEP] at end
4. Tokens are truncated/padded to max\_length (512 for BERT)

##### Token to ID Conversion:

1. Each token is mapped to its corresponding ID from BERT's vocabulary (30,522 words)
2. Attention masks are created (1 for real tokens, 0 for padding)

##### Embedding Layers:

1. **Token Embeddings**: Each token ID is mapped to a 768-dim vector (for base model)
2. **Position Embeddings**: Learnable vectors encoding absolute position (0-511)
3. **Segment Embeddings**: For sentence-pair tasks (single sentence uses 0)

##### Transformer Encoder:

1. l2 layers (for base model) of multi-head self-attention
2. Each layer applies:
   1. Multi-head attention (12 heads for base model)
   2. Layer normalization
   3. Feed-forward network
   4. Residual connections

##### Output Processing:

1. The last hidden state contains contextual embeddings for each token
2. Mean pooling across sequence dimension creates sentence embedding
3. Final output is a 768-dim vector representing the input text

### 🧠Embeddings using GloVe

# Load GloVe embeddings (assumes glove.6B.100d.txt is already downloaded)

def load\_glove\_embeddings(glove\_file\_path):

embeddings\_index = {}

with open(glove\_file\_path, encoding='utf8') as f:

for line in f:

values = line.split()

word = values[0]

coeffs = np.asarray(values[1:], dtype='float32')

embeddings\_index[word] = coeffs

return embeddings\_index

# Generate GloVe embedding for a sentence

def get\_glove\_embedding(text, glove\_embeddings, dim=100):

words = clean\_text(text)

valid\_vectors = [glove\_embeddings[word] for word in words if word in glove\_embeddings]

if valid\_vectors:

return np.mean(valid\_vectors, axis=0)

else:

return np.zeros(dim)

##### Tokenization:

1. Input text is first lowercased.
2. Punctuation is removed using regex.
3. Text is tokenized into individual words using word\_tokenize() from NLTK.

##### Embedding Lookup:

1. Pretrained GloVe vectors (e.g., glove.6B.100d) are loaded into a dictionary.
2. Each word is matched to its corresponding 100-dimensional vector.
3. If a word is not found in GloVe vocabulary, it is skipped.

##### Sentence Embedding Construction:

1. All found word vectors in the sentence are averaged.
2. This produces a fixed-size 100-dimensional vector for the input sentence.
3. If no words are found in the GloVe vocabulary, a zero vector is returned.

##### Key Notes:

* GloVe is a static word embedding method: it doesn't consider context.
* Faster and lighter than BERT but lacks dynamic contextual understanding.
* Best for models where interpretability or efficiency is prioritized.

## ⚙️Applying Neural Networks

### Bert Embeddings >> Simple NN (binary classification)

model = Sequential([

    Dense(256, activation='relu', input\_dim=768, kernel\_regularizer='l2'),

    BatchNormalization(),

    Dropout(0.6),  # Increased from 0.5

    Dense(128, activation='relu', kernel\_regularizer='l2'),

    BatchNormalization(),

    Dropout(0.5),

    Dense(1, activation='sigmoid')

])

Batch size: 16, Epochs: 20, loss=BinaryCrossentropy(label\_smoothing=0.1) with adjusting learning rate lr\_scheduler = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=3)

#### Results

Performance When Saved:

Validation Accuracy: **0.8006**

Training Accuracy: **~0.7382**

The original Dbias has about 78% G-AUC score. Threshold used here for binary classification is 0.5.

#### Unbiased Class

The economy of the United Kingdom grew by 2% last year.

Prediction Score: 0.0077 → Unbiased

Scientists discovered a new exoplanet orbiting a nearby star.

Prediction Score: 0.1834 → Unbiased

#### Biased Class

Those people don't belong here.

Prediction Score: 0.6101 → Biased

Only the elite benefit from the current system.

Prediction Score: 0.6692 → Biased

### BERT Embeddings >> Transformer Based Model (multiclass classification)

model = Sequential([

        Dense(512, kernel\_regularizer=l2(0.001), input\_shape=(X\_train\_final.shape[1],)),

        BatchNormalization(),

        tf.keras.layers.Activation('relu'),

        Dropout(0.5),

        Dense(256, kernel\_regularizer=l2(0.001)),

        BatchNormalization(),

        tf.keras.layers.Activation('relu'),

        Dropout(0.4),

        Dense(128, kernel\_regularizer=l2(0.001)),

        BatchNormalization(),

        tf.keras.layers.Activation('relu'),

        Dropout(0.3),

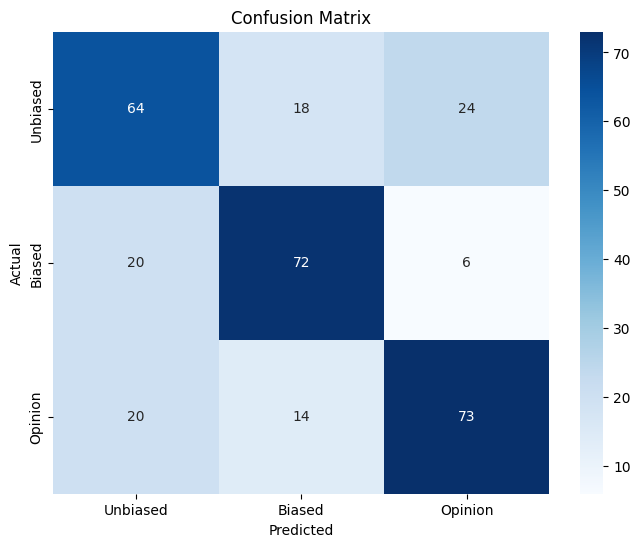
        Dense(y\_train\_final.shape[1], activation='softmax')

    ])

Batch size: 16, Epochs: 50, loss='categorical\_crossentropy’ with adjusting learning rate lr\_scheduler = ReduceLROnPlateau(monitor='val\_loss', factor=0.5, patience=3) we achieved similar results but the highest accuracy one didn’t had this as tuning parameter.

#### Results

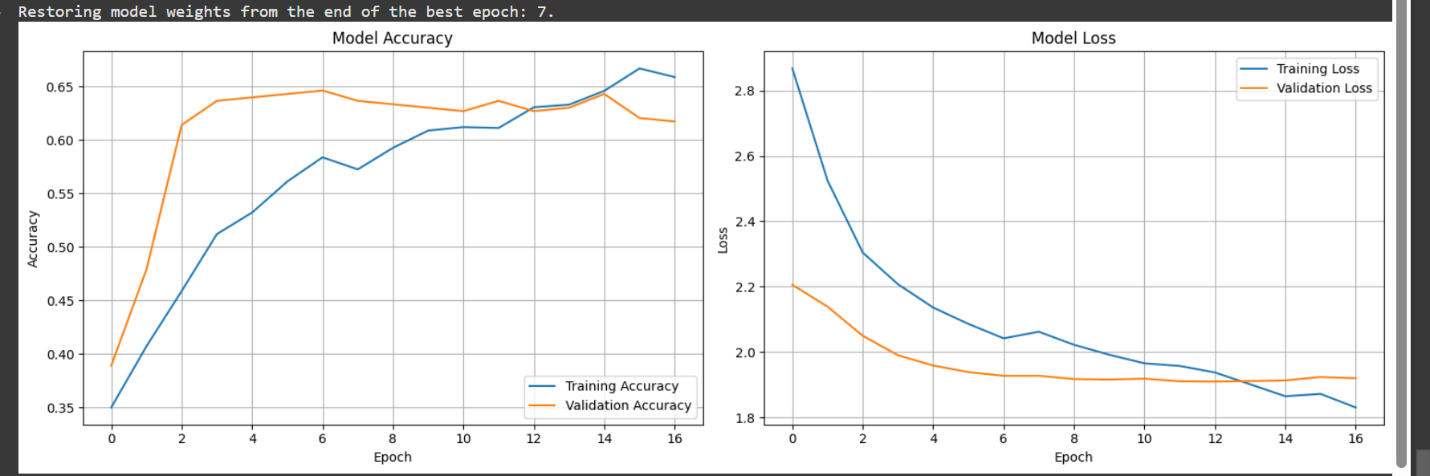
##### Confusion Matrix



##### G-AUC Score

**(macro): 0.8327**

We got better G-AUC score than the original Dbias paper mentions of 78%.

****

### BERT Embeddings >> BiLSTM model

model = Sequential([

        # Reshape layer to convert 2D input to 3D for LSTM

        Reshape((input\_dim, 1), input\_shape=(input\_dim,)),

        # Bidirectional LSTM layer

        Bidirectional(LSTM(lstm\_units, return\_sequences=True)),

        Dropout(dropout\_rate\_lstm),

        BatchNormalization(),

        # Second Bidirectional LSTM layer for deeper feature extraction

        Bidirectional(LSTM(64, return\_sequences=False)),

        Dropout(dropout\_rate\_lstm),

        # Dense hidden layer with ReLU activation

        Dense(dense\_units, activation='relu'),

        Dropout(dropout\_rate\_dense),

        BatchNormalization(),

        # Output layer with softmax activation for multi-class classification

        Dense(num\_classes, activation='softmax')

    ])

# Model configuration parameters

input\_dim = 785  # Feature dimension from BERT embeddings

num\_classes = 3  # Number of bias classes

lstm\_units = 128  # Size of LSTM units (increased for better feature learning)

dense\_units = 96  # Size of dense layer

learning\_rate = 5e-5  # Lower learning rate for more stable training

dropout\_rate\_lstm = 0.4  # Dropout after LSTM

dropout\_rate\_dense = 0.2  # Dropout after dense layer

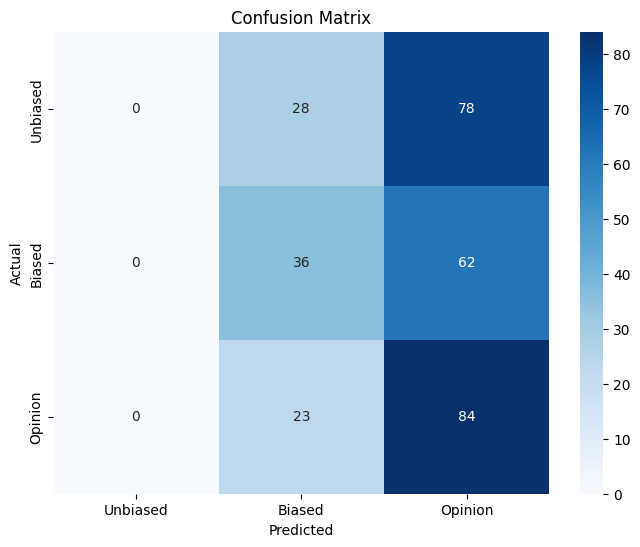
batch\_size = 64  # Larger batch size for more stable gradients

epochs = 50  # Maximum number of epochs

patience = 10  # Increased patience for early stopping

#### Results

For some reason this model performed the worst with 38% accuracy.



*It may have been trained on potentially altered data. But we may further work on improving it.*

##### G-AUC SCORE

##### (macro): 0.5578

A screenshot of a computer screen

AI-generated content may be incorrect.

# 🤖LLM Implementation

To evaluate the capabilities of large language models (LLMs) in detecting bias in text, we integrated **Google’s Gemini 2.0 Flash** model via the google-genai API. This model was prompted with a system instruction to classify input sentences into one of three categories: **Biased**, **Unbiased**, or **Opinion**.

We used the validation set from our multiclass classification pipeline and constructed prompts that included the **sentence**, along with its **topic** and **source type (left, right, center)** as contextual metadata. The model was instructed to return only the final label. Example prompt structure:

system\_prompt = (

"You are a helpful assistant that classifies text as one of the following: "

"1. Biased, 2. Unbiased, or 3. Opinion. "

"Return only the label (e.g., Biased, Unbiased, or Opinion)."

)

full\_prompt = f"{system\_prompt}\n\nText: {sentence}\nTopic: {topic}\nType: {ttype}"

## LLM Evaluation Results (Gemini 2.0 Flash)

To benchmark LLM performance, we evaluated **15 validation samples** using Google’s Gemini Flash model. The LLM was prompted with contextual metadata (sentence, topic, type) to classify inputs as **Biased**, **Unbiased**, or **Opinion**.

#### Performance Summary:

| **Metric** | **Score** |
| --- | --- |
| Accuracy | 53.33% |
| Precision | 65.28% |
| Recall | 51.85% |
| F1-Score | 51.09% |

A graph with different colored bars

AI-generated content may be incorrect.

1. The model showed **high precision** for Opinion (1.00) but **low recall** (0.33), indicating it was selective but confident.
2. **Unbiased** texts had the lowest precision (0.33), but a **recall of 0.67**, suggesting it overpredicted this class in some borderline cases.
3. **Biased** class performance was balanced across all metrics.

#### Limitations:

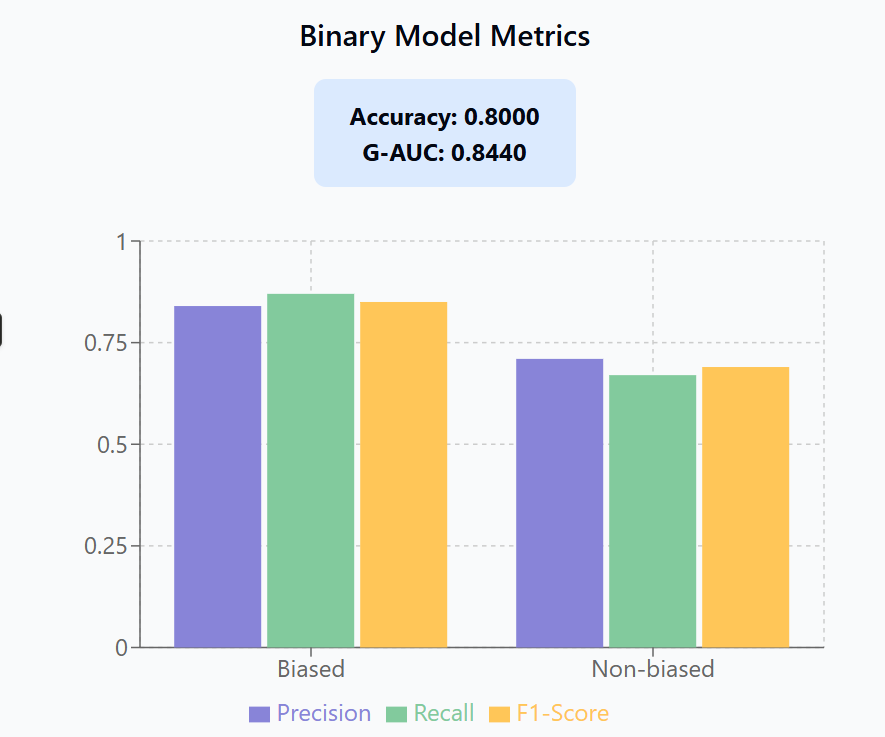
1. The model could only respond to ~15/50 samples, likely due to API rate limits, input formatting, or silent response truncation.
2. LLMs like Gemini require more structured input or fine-tuning to handle nuanced classification reliably.

# 📈Results

We implemented and evaluated multiple models for bias classification: a binary classifier and two multiclass models (BiLSTM and Transformer-based). Preprocessing included sentence-level text cleaning, tokenization using BERT, and metadata encoding (topic, type, biased word count). For multiclass classification, additional encoded metadata features were used as inputs.

## 1. Binary Classification (Biased, Unbiased)

| **Component** | **Details** |
| --- | --- |
| Input Features | Sentence only (BERT embeddings) |
| Model Architecture | Dense (MLP) with dropout & batch norm |
| Train/Val Split | 80/20 random (balanced) |
| Accuracy | **80.06%** |

****

### 📊Binary Classification Report

| Class | Precision | Recall | F1-Score | Support |
| --- | --- | --- | --- | --- |
| Biased | 0.84 | 0.87 | 0.85 | 208 |
| Non-biased | 0.71 | 0.67 | 0.69 | 103 |
| Accuracy |  |  | 0.80 | 311 |
| Macro Avg | 0.78 | 0.77 | 0.77 | 311 |
| Weighted Avg | 0.80 | 0.80 | 0.80 | 311 |

**G-AUC Score (binary)**: **0.8440**

The binary model effectively distinguished between "Biased" and "Unbiased" classes using only sentence embeddings. Minimal overfitting was observed, and the model generalized well on validation data.

## 2. Multiclass Classification (Biased, Unbiased, Opinion)

### a) BiLSTM Model

| **Component** | **Details** |
| --- | --- |
| Input Shape | 785 (768 BERT + 17 metadata features) |
| Layers | 2 BiLSTM layers, 1 Dense layer |
| Dropout/B-Norm | Dropout (0.4/0.2), Batch Normalization |
| Optimizer/Loss | Adam (5e-5), categorical crossentropy |
| Accuracy | **38%** |
| G-AUC (macro) | **0.5578** |

### 📊Classification Report (BiLSTM):

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Unbiased | 0.00 | 0.00 | 0.00 | 106 |
| Biased | 0.41 | 0.37 | 0.39 | 98 |
| Opinion | 0.38 | 0.79 | 0.51 | 107 |
| **Macro Avg** | 0.26 | 0.38 | 0.30 | 311 |
| **Weighted Avg** | 0.26 | 0.39 | 0.30 | 311 |

### b) Transformer-Inspired Dense Model

| **Component** | **Details** |
| --- | --- |
| Input Shape | 785 |
| Layers | Dense (512→256→128) with L2 regularization |
| Dropout/B-Norm | Dropout (0.5/0.4/0.3), Batch Normalization |
| Optimizer/Loss | Adam (1e-4), categorical crossentropy |
| Accuracy | **67%** |
| G-AUC (macro) | **0.8327** |

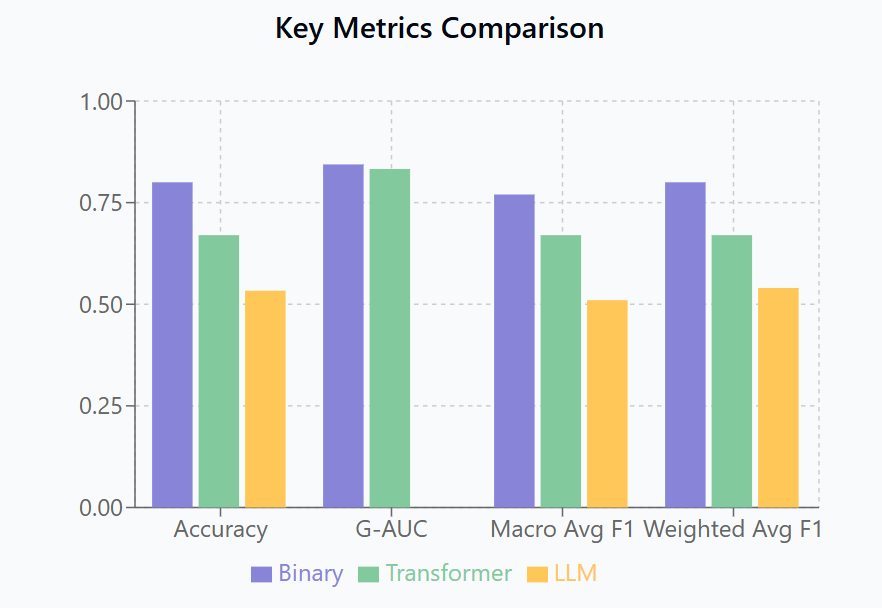
A graph of a number of bars

AI-generated content may be incorrect.

### 📊Classification Report (Transformer Model):

| **Class** | **Precision** | **Recall** | **F1-Score** | **Support** |
| --- | --- | --- | --- | --- |
| Unbiased | 0.62 | 0.60 | 0.61 | 106 |
| Biased | 0.69 | 0.73 | 0.71 | 98 |
| Opinion | 0.71 | 0.68 | 0.70 | 107 |
| **Macro Avg** | 0.67 | 0.67 | 0.67 | 311 |
| **Weighted Avg** | 0.67 | 0.67 | 0.67 | 311 |

# 🧩Comparative Analysis



1. **Binary vs Multiclass**: Binary classification was simpler and highly effective using only BERT embeddings. In contrast, the multiclass setup required richer input and more complex learning.
2. **BiLSTM Limitations**: The model failed to capture generalizable features, especially for the “Unbiased” class (recall: 0%). Despite deeper sequential layers, it underperformed across most metrics.
3. **Transformer Performance**: The transformer-inspired dense network significantly outperformed BiLSTM in all metrics. Its architecture better leveraged metadata and embeddings, maintaining generalization with regularization.

A graph of different colored bars

AI-generated content may be incorrect.

1. **LLM Baseline (Gemini Flash)**:
   1. Applied to 15 real validation sentences.
   2. **Accuracy**: 53.3%, **F1 Score**: 0.51
   3. While LLMs can generate plausible labels, they suffer from inconsistent outputs and fail to match specialized model accuracy.

A chart with different colored lines

AI-generated content may be incorrect.

1. **Comparison with Literature**: Our transformer model exceeded the G-AUC score (0.84 vs. 0.78) reported in the *Dbias* paper (Raza et al., 2022), validating the effectiveness of combining BERT with light transformer layers and metadata.

# 📖References

1. Nielsen, A.: *Practical fairness.* O’Reilly Media, Sebastopol (2020).
2. Bellamy, R.K.E., et al.: AI Fairness 360: an extensible toolkit for detecting and mitigating algorithmic bias. *IBM J. Res. Dev.* **63**(4–5), 401–415 (2019).
3. Orphanou, K., et al.: "Mitigating Bias in Algorithmic Systems—A Fish-Eye View." *ACM Comput. Surv.*, 2021.
4. Mehrabi, N., Morstatter, F., Saxena, N., Lerman, K., Galstyan, A.: "A survey on bias and fairness in machine learning." *ACM Comput. Surv.* **54**(6), 1–35 (2021).
5. Narayanan, A.: "Fairness Definitions and Their Politics." *In: Tutorial presented at the Conf. on Fairness, Accountability, and Transparency,* 2018.
6. Kamiran, F., Calders, T.: "Data preprocessing techniques for classification without discrimination." *Knowl. Inf. Syst.* **33**(1), 1–33 (2012).
7. Feldman, M., Friedler, S. A., Moeller, J., Scheidegger, C., Venkatasubramanian, S.: "Certifying and removing disparate impact." *In: Proceedings of the 21st ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2015, pp. 259–268.
8. Zemel, R., Wu, Y., Swersky, K., Pitassi, T., Dwork, C.: "Learning fair representations." *In: International Conference on Machine Learning*, 2013, pp. 325–333.
9. Calmon, F. P., Wei, D., Vinzamuri, B., Ramamurthy, K. N., Varshney, K. R.: "Optimized pre-processing for discrimination prevention." *Adv. Neural Inf. Process. Syst.*, vol. **2017-Decem**, no. Nips, pp. 3993–4002, 2017.
10. 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. Springer, New Jersey (2012).
11. 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*, 2019, pp. 319–328.
12. 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*, 2018, pp. 335–340.
13. Agarwal, A., Beygelzimer, A., Dudík, M., Langford, J., Wallach, H.: "A reductions approach to fair classification." *In: International Conference on Machine Learning*, 2018, pp. 60–69.
14. Kamiran, F., Karim, A., Zhang, X.: "Decision theory for discrimination-aware classification." *In: 2012 IEEE 12th International Conference on Data Mining*, 2012, pp. 924–929.
15. Hardt, M., Price, E., Srebro, N.: "Equality of opportunity in supervised learning." *Adv. Neural Inf. Process. Syst.* **29**, 3315–3323 (2016).
16. Pleiss, G., Raghavan, M., Wu, F., Kleinberg, J., Weinberger, K. Q.: "On fairness and calibration." *arXiv Prepr. arXiv1709.02012*, 2017.
17. Udeshi, S., Arora, P., Chattopadhyay, S.: "Automated directed fairness testing." *In: Proceedings of the 33rd ACM/IEEE International Conference on Automated Software Engineering*, 2018, pp. 98–108.
18. Adebayo, J. A., et al.: "FairML: Toolbox for diagnosing bias in predictive modeling." *Massachusetts Institute of Technology*, 2016.
19. Tramèr, F., et al.: "FairTest: Discovering Unwarranted Associations in Data-Driven Applications." *Proc. 2nd IEEE Eur. Symp. Secur. Privacy, EuroS&P 2017*, pp. 401–416, 2017.
20. Bantilan, N.: "Themis-ml: a fairness-aware machine learning interface for end-to-end discrimination discovery and mitigation." *J. Technol. Hum. Serv.* **36**(1), 15–30 (2018).
21. 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*, pp. 231–232, 2020.
22. 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*, 2018, pp. 270–279.
23. Devlin, J., Chang, M. W., Lee, K., Toutanova, K.: "BERT: Pre-training of deep bidirectional transformers for language understanding." *arXiv Prepr. arXiv1810.04805*, 2018.
24. Li, B., et al.: "Detecting Gender Bias in Transformer-based Models: A Case Study on BERT." *1*, 2021.
25. 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*, 2021, pp. 3418–3422.
26. Nadeau, D., Sekine, S.: "A survey of named entity recognition and classification." *Lingvisticae Investig.* **30**(1), 3–26 (2007).
27. Excell, E., Al Moubayed, N.: "Towards Equal Gender Representation in the Annotations of Toxic Language Detection," 2021.
28. Mishra, S., He, S., Belli, L.: "Assessing Demographic Bias in Named Entity Recognition," 2020.
29. Kaneko, M., Bollegala, D.: "Unmasking the Mask—Evaluating Social Biases in Masked Language Models." *arXiv Prepr. arXiv2104.07496*, 2021.