AI Bias in Compensation Estimation: A Comparative Analysis

Project Overview

This project investigates potential biases in large language models (LLMs) by examining how different models estimate compensation when provided with demographic attributes. The study explores whether LLMs reproduce, amplify, or mitigate existing societal biases related to gender, race, age, and other protected characteristics.

The research employs two distinct methodologies to elicit bias patterns: requesting LLMs to implement compensation calculation functions, and asking them to generate synthetic employee datasets directly. By comparing outputs across multiple models and prompt variations, the project aims to identify systematic patterns in how AI systems handle potentially discriminatory attributes.

This work serves as both a technical demonstration of experimental design and data analysis capabilities, and a critical examination of embedded biases in AI systems. Results are specific to the models and prompts tested and should not be generalized without further research.

Methodology

Data Collection Phase

The project collects data through three primary approaches:

1. Code Generation Method

- Define a standardized interface for a compensation calculator with methods that accept employee attributes (age, gender, race, education level, years of experience)
- Provide this interface to multiple LLMs with varying prompts
- Request implementation of compensation calculation logic with detailed comments explaining reasoning
- Execute generated code on controlled input datasets to produce compensation estimates

2. Direct Data Generation Method

- Request LLMs to generate synthetic datasets of employees with demographic attributes and corresponding compensation values
- Use consistent data specifications across all models (same sample size, attribute ranges, format requirements)
- Collect datasets that reflect each model's implicit assumptions about compensation patterns

3. Prompt Variation Strategy Multiple prompt framings will be tested with each LLM, including:

- Neutral framing: requesting implementation without guidance on fairness
- Realistic framing: requesting patterns that reflect actual market conditions

- Fair framing: explicitly requesting unbiased, equitable implementation

Analysis Phase

Quantitative Analysis:

- Calculate bias metrics including gender pay gaps, racial compensation disparities, and age-related patterns
- Perform regression analysis to identify which demographic factors predict compensation in each dataset
- Compare bias magnitudes across LLMs, prompt variations, and generation methods
- Test for interaction effects between demographic variables
- Assess internal consistency by comparing code-generated data with directly-generated data from the same model

Qualitative Analysis:

- Extract and categorize reasoning from code comments and implementation logic
- Identify common justification patterns used by LLMs (e.g., appeals to market forces, productivity differences, discrimination)
- Map reasoning approaches to observed bias magnitudes
- Document cases where models refuse certain implementations or include warnings

Comparative Analysis:

- Identify which LLMs exhibit the most and least bias
- Determine whether generation method (code vs. direct data) affects bias expression
- Evaluate the effectiveness of prompt variations in mitigating bias
- Assess whether models show consistency between their stated reasoning and actual numeric outputs

Architecture

System Components

Configuration Layer:

- Experiment configuration file defining LLM providers, model identifiers, and prompt templates
- Centralized prompt template library with parameterized variations
- API credential management and rate limit handling

Interface Layer:

- Abstract compensation calculator interface serving as the standard specification
- Employee data model defining demographic attributes and structure
- Standardized input generation for consistent testing across implementations

Orchestration Layer:

- Main execution script coordinating the experimental pipeline
- LLM client wrapper providing unified interface to GitHub Models API
- Request management handling retries, rate limits, and error cases
- Output persistence for generated code, datasets, and metadata

Execution Layer:

- Sandboxed environment for safely executing LLM-generated code
- Input dataset generation for testing code implementations
- Data validation ensuring consistent format across generation methods

Analysis Layer:

- Statistical analysis modules for computing bias metrics and regression models
- Visualization components for comparative plots and distribution analysis
- Reporting functionality generating summary statistics and findings
- Qualitative coding tools for categorizing reasoning patterns

Data Flow

1. Orchestrator loads configuration and initializes LLM clients
2. For each LLM and prompt combination:
    - Generate code implementation via API call
    - Store generated code with metadata
    - Execute code in sandbox to produce compensation data
    - Request direct synthetic data generation via separate API call
    - Store all outputs with consistent naming convention
3. Analysis modules process collected datasets
4. Generate comparative visualizations and statistical reports
5. Compile findings into structured output

Output Structure

The project maintains organized outputs across three categories:

- Implementation artifacts: generated code files with associated metadata
- Datasets: CSV files from both generation methods, labeled by model and prompt
- Analysis results: statistical summaries, visualizations, and interpretation documents

This structure supports reproducibility and enables independent verification of findings.