CSC398 - Utilization of Instruct Lab for Bias Detection

Authors: Shaarif Ali Syed, Asser Ismail, Remy Zazo, Zelimir Stajic, Belal Shrief

Executive Summary:

This project uses RedHat x IBM InstructLab to fine-tune our Bias Detection Model. Using taxonomies across multiple bias categories, the model identifies bias in text, highlights the relevant content, and explains why it is biased, which helps users understand what to avoid.

Project Background:

As language models become more widely used in writing, analysis and decision making, there are concerns about bias in AI-generated text as they have become increasingly influential all around us. Bias in text can lead to misinformation, reinforce stereotypes and influence how information is interpreted. Detecting bias is challenging because it is often quite subtle and depends heavily on the context it's given.

Traditional approaches to training models rely on large datasets, but these datasets are unstructured and may not effectively teach the model how to properly reason with bias. InstructLab provides an alternative approach by allowing developers to define structured taxonomies and generate synthetic data from a small number of examples. This makes it possible to guide a model's behaviour more directly without needing massive datasets or full retraining. This project builds on that idea by applying taxonomy-driven training specifically to the problem of bias detection.

Problem Definition:

Bias in text and AI systems is difficult to detect because it is often subtle and context-dependent. A statement may appear neutral on the surface, but it can still contain underlying assumptions or stereotypes. Most language models that are used for general purposes are not specifically trained to recognize these patterns consistently and effectively. In the early stages of this project, we found that simply providing the model with definitions of bias was not enough. While the model could explain what bias is, it struggled to apply that understanding to real examples. This highlighted the need for a more structured approach that focuses on how the model reasons about text rather than just what it knows.

Proposed Solution:

To address this problem, we used the Red Hat x IBM InstructLab framework to fine-tune our bias detection model. Instead of relying on large unstructured datasets, we defined structured taxonomies across multiple bias categories, which include political, gender, ageism, ethical, marketing, and research bias. These taxonomies were used to generate synthetic training data, which allowed us to scale from a small set of examples into a larger dataset. The generated data was then used to fine-tune a Granite-7B base model. As a result, the system can take a user input and determine whether bias is present, categorize the type of bias it has and provide an explanation of its reasoning.

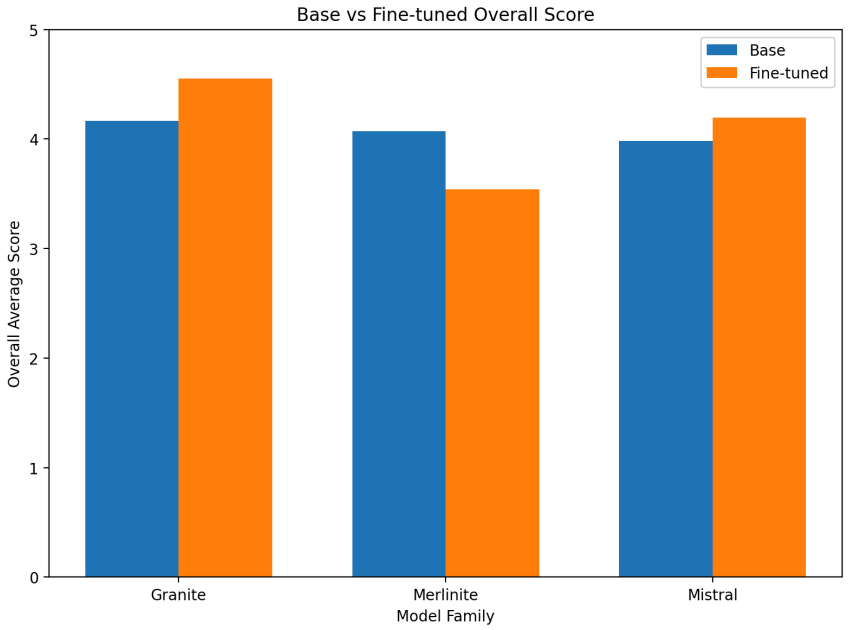
Results:

MT-Bench Style Results:

To assess the performance of the fine-tuned model, we focused on evaluating both bias classification accuracy and the quality of generated explanations. The results highlight the impact of the fine-tuning across different model architectures.

The first figure shows the overall average scores for each model before and after fine-tuning. Fine-tuning led to **improvements in Granite and Mistral**, but a **decline in Merlinite**.

- **Granite** improved from approximately 4.2 to 4.5, showing the most significant gain among all models.
- **Mistral** also improved modestly from around 4.0 to 4.2.
- **Merlinite**, however, dropped from roughly 4.1 to 3.5 after fine-tuning.



This suggests that fine-tuning was effective for some architectures but not universally beneficial. The decline in Merlinite indicates that the fine-tuning process or dataset may not have aligned well with its underlying structure.

Kaggle Dataset Results

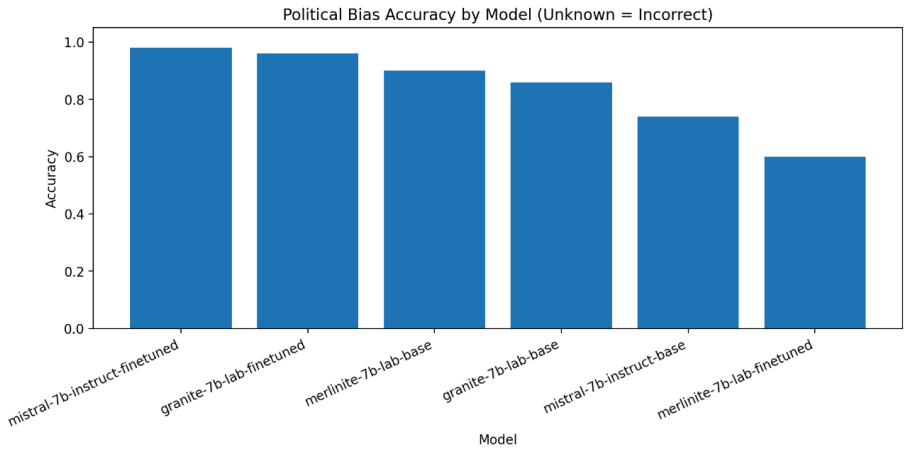
The performance of the evaluated models was assessed across two bias detection tasks: **political bias** and **sexism**, using both **accuracy** and **F1 score** as evaluation metrics. Additionally, any outputs labelled as *"unknown"* were treated as incorrect predictions, making the evaluation stricter and more reflective of real-world usability.

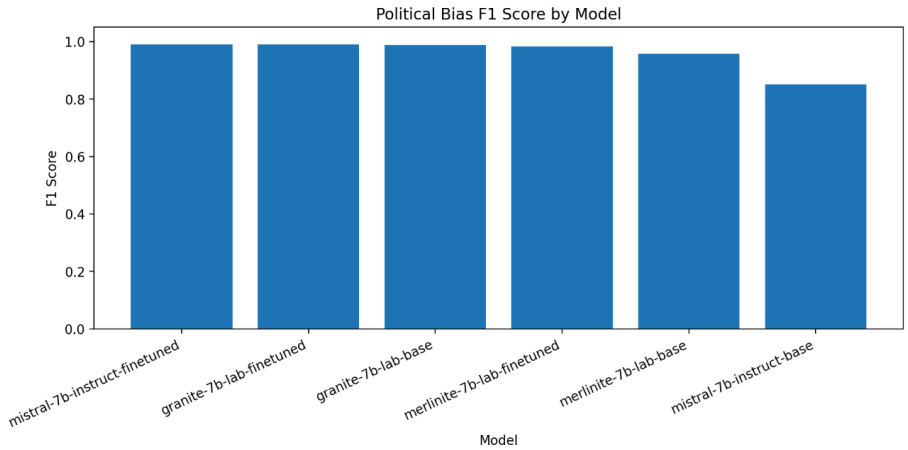
Political Bias Detection:

Across all models, performance on political bias detection was consistently high. The best performing models were **mistral-7b-instruct-finetuned** and **granite-7b-lab-finetuned**, both achieving near perfect accuracy (≈0.98-0.99) and F1 scores close to 1.0. This indicates that fine-tuning had a significant positive impact on the models' ability to detect political bias.

Base models such as **granite-7b-lab-base** and **merlinite-7b-lab-base** also performed reasonably well, achieving accuracy scores in the range of 0.85-0.90. However, their performance was noticeably lower than their fine-tuned counterparts, highlighting the importance of task-specific training.

Interestingly, while **mistral-7b-instruct-base** achieved a slightly lower accuracy (~0.74), its F1 score remained relatively high (~0.95). This suggests that while the model may have struggled with certain classifications (possibly due to "unknown" outputs), it still maintained a strong balance between precision and recall when making confident predictions.



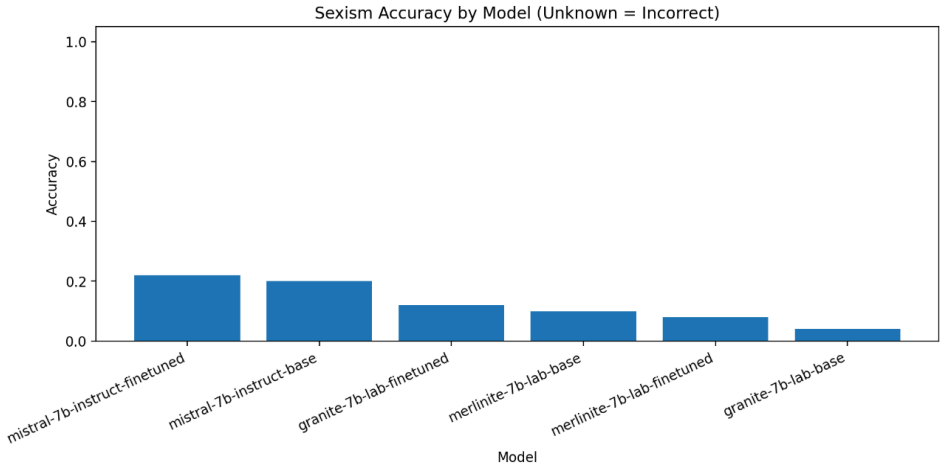


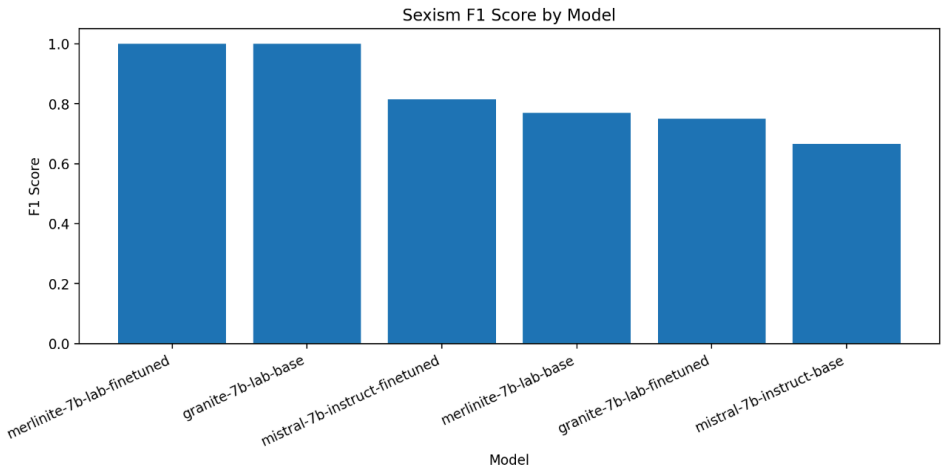
Sexism Detection:

In contrast, sexism detection proved to be significantly more challenging for all models. Accuracy scores were substantially lower across the board, with the best performing model (**mistral-7b-instruct-finetuned**) achieving only around 0.22 accuracy.

This sharp drop in accuracy suggests that models frequently produced incorrect or “unknown” classifications when dealing with sexism-related content. However, the F1 scores tell a more nuanced story. For example, **merlinite-7b-lab-finetuned** and **granite-7b-lab-base** achieved F1 scores close to 1.0, despite relatively low accuracy.

Additionally, base models such as **granite-7b-lab-base** and **mistral-7b-instruct-base** performed poorly in terms of accuracy (as low as ~0.04), further emphasizing the difficulty of this task without fine-tuning.





Technical Implementation:

The technical implementation of this project is structured across three phases: Taxonomy design, synthetic data generation (SDG), and model training & evaluation.

Taxonomy design:

Taxonomy design was the foundation of the project, and how we structured the InstructLab taxonomies determined how the rest of the phases went. InstructLab offers two types of taxonomies: knowledge taxonomies, which teach the model factual information, and skills taxonomies, which teach the model how to perform a task. At the early stages of the project, we experimented with knowledge-based taxonomies to teach the model the definitions of the different types of biases. However, we found that this approach only allowed the model to output definitions rather than apply reasoning to real examples and flag underlying bias. We then tried grounded compositional skills economies, which are context dependent and require the model to analyze a provided input as needed, rather than recalling definitions. Our taxonomies were in the form of qna.yaml seed files that guided the model to reason about the presence of bias, the type of bias, where in the text it appears, and the reasoning behind why it is considered biased.

SDG:

The SDG pipeline was used to scale our small set of skill based taxonomy examples into a large structured training dataset. The simple pipeline was chosen as used for SDG, which is optimized for generating accurate Q&A pairs directly from compositional skills taxonomies. The pipeline goes through three stages to generate the synthetic data: generation, filtering, and validation. The generation stage consists of the production of thousands of synthetic questions that are modeled after the taxonomy structure. The filtering stage then passes these questions into an internal judge model (merlinite-7b) which gets rid of low scoring or low quality questions. Lastly, the pipeline generates answers for the high scoring questions and evaluates them against the original seed logic.

Model Training & Evaluation:

The filtered SDG was used to fine-tune 3 different models: **granite-7b-lab, mistral-7b-instruct, merlinite-7b-lab**. Two assessment approaches were used to evaluate our fine tuned model. The first was an MT-Bench style evaluation using a custom dataset of biased and unbiased statements, where the model was prompted to classify the input and explain its reasoning behind the classification. These responses were then scored by GPT as an external judge based on classification accuracy and explanation quality. The second approach was a labeled Kaggle dataset containing real world bias annotation categories such as sexism and political bias. Model predictions were compared against the ground truth labels to measure accuracy, and that was its evaluation criterion.

MT-Bench Style Evaluation Criteria:

To evaluate model performance, we consider three key aspects: **assessment correctness**, **categorization correctness**, and **reasoning quality**.

1. **Assessment Correctness** measures whether the model correctly identifies if a given statement contains bias or not.
2. **Categorization Correctness** evaluates whether the model correctly classifies the type of bias (e.g., ageism, sexism, political bias).
3. **Reasoning Quality** assesses how well the model explains its decision, including clarity, coherence, and justification.

Assessment Correctness

All models maintained relatively high scores in assessment correctness:

- Granite saw a slight improvement
- Mistral remained relatively stable
- Merlinite experienced a small decrease

This indicates that base models already perform well in identifying whether bias exists, and fine-tuning does not drastically impact this capability.

Categorization Correctness

Categorization correctness consistently received the lowest scores across all models, highlighting it as the most challenging evaluation dimension. Despite this, it demonstrated the most significant gains from fine-tuning.

- Granite improved significantly after fine-tuning (from ~3.2 to ~4.0)
- Mistral showed a modest increase
- Merlinite declined further, reaching the lowest score (~2.3)

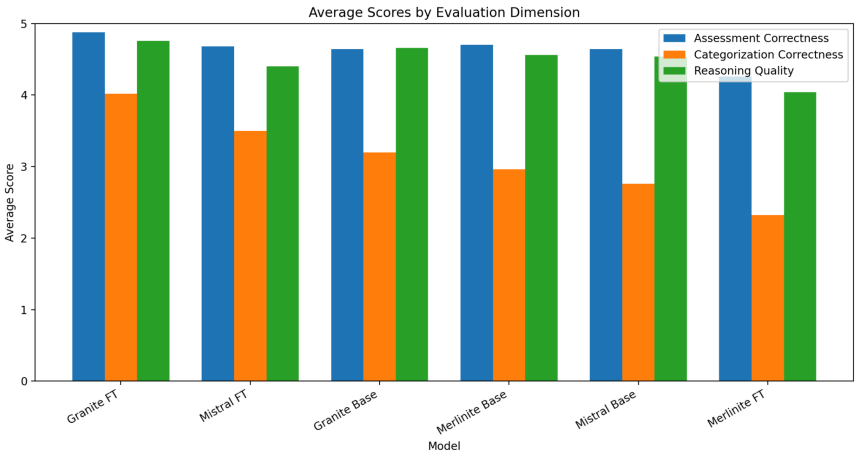
This highlights that fine-tuning can substantially improve classification performance when aligned properly, but can also degrade performance if not well-tuned.

Reasoning Quality

Reasoning quality scores were generally strong across all models:

- Granite improved slightly after fine-tuning (~4.6 to ~4.7)
- Mistral remained relatively stable
- Merlinite again showed a noticeable drop

This suggests that while reasoning ability is relatively robust in base models, fine-tuning can either enhance or harm this capability depending on the model.



Ethical Implications:

There are several ethical considerations that are important to acknowledge when developing a bias detection model. Bias, discrimination, and stereotypes are all sensitive social concepts that the model addresses, which means that the model's outputs can have real consequences for how people interpret and revise their writing.

False positives are a big risk with bias detection models, as the model might incorrectly flag content as bias, especially in examples where bias is very subtle and heavily dependent on context. This model's intended use is for advisory feedback, over reliance on the model is another ethical concern, as people may treat the output as factual rather than advisory.

Another ethical consideration is that the model is designed to provide neutral explanations as much as possible within the capabilities of the developers to ensure the model is as unbiased as possible without accusatory or moralizing language. The use of SDG also raises questions about whether biases may have been introduced during the generation process itself by the Merlinite-7B judge model that would then bleed into the fine-tuned Granite-7B model.

Recommendations:

While the current system demonstrates promising results in detecting bias, there are several areas where the project can be extended and improved.

One key direction is expanding the evaluation methodology. In this project, we relied on a combination of manually curated prompts (inspired by MT-Bench) and labelled datasets from Kaggle. Future work could incorporate additional benchmark datasets, adversarial test cases, and more diverse real-world text sources to better evaluate robustness.

Another important area is prompt engineering. During experimentation, we observed that model performance was highly sensitive to prompt design. Some prompts resulted in weaker classifications, while others significantly improved accuracy and explanation quality. This suggests that a systematic exploration of prompt variations, prompt tuning strategies, or even automated prompt optimization could lead to more consistent and improved performance.

Additionally, the choice of evaluation model (i.e., the LLM used as a judge) can impact results. In this project, we used a single scoring model, however, future work could explore ensemble evaluation (multiple judges), alternative models to reduce bias and improve reliability in scoring.

Finally, it is worth noting that InstructLab, the framework used in this project, is expected to be deprecated and replaced by a newer system. Future work could involve migrating this pipeline to the updated framework once available, allowing for improved tooling, better model alignment workflows, and potentially more efficient training and evaluation processes.

Reference:

Bai, G., Liu, J., Bu, X., He, Y., Liu, J., Zhou, Z., Lin, Z., Su, W., Ge, T., Zheng, B., & Ouyang, W. (2024). MT-Bench-101: A fine-grained benchmark for evaluating large language models in multi-turn dialogues. arXiv. https://doi.org/10.48550/arXiv.2402.14762

OpenAI. (2026). *ChatGPT (GPT-5.3)* [Large language model]. https://chat.openai.com/

Red Hat, Inc. & IBM. (2024). *InstructLab documentation*. https://docs.instructlab.ai/

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., & Stoica, I. (2023). *Judging LLM-as-a-judge with MT-Bench and Chatbot Arena*. arXiv. https://doi.org/10.48550/arXiv.2306.05685