**Summary of our Work**

**Yakov Schory**,**Lia Roiyhvarg** ResearchGenAI

**David Aslanyan, Michael Kulikov** CodeCrusaders

The paper "Evaluating Large Language Models Trained on Code" introduced Codex (a GPT-based language model fine-tuned on publicly available code). The study evaluated Codex’s ability to generate functional code from docstrings and compared its performance with previous models like GPT-3 and GPT-J on the HumanEval benchmark — a new dataset created to assess the correctness of code by using unit tests. Evaluations were performed using the pass@k metric that uses unit tests to evaluate code correctness.

The second paper, "Capturing Failures of Large Language Models via Human Cognitive Biases," delves into the challenge of mitigating biases in LLMs. It examines various cognitive biases, such as the framing effect, anchoring effect, and availability heuristic, and evaluates how models like GPT-3 and Codex are influenced by these biases. The paper demonstrates how these biases manifest differently in AI models, highlighting the importance of addressing them to improve the accuracy and fairness of LLM-generated outputs.

Incorporating insights from these papers, our solution integrates key findings into a practical framework. We utilized the pass@k metric to compare the performance of the GPT-2 model with a fine-tuned SFT (supervised fine tuning) model. Our objective was to generate optimized outputs with reduced cognitive biases. To achieve this, we employed the HumanEval dataset as our primary source for programming problems. By leveraging natural language processing algorithms, the tool aims to enhance the accuracy and functionality of AI-generated code while addressing the biases identified in the second paper and improving upon the code generation techniques discussed in the first paper. This approach not only refines prompts but also evaluates outputs against a benchmark dataset, ensuring that the final product benefits from both improved performance and reduced cognitive biases.

Our Work Process Findings:

**Are Large Language Models Truly Free from Cognitive Biases?**

Large Language Models (LLMs), such as GPT-2 and Llama, have demonstrated remarkable capabilities in generating human-like text and code. However, these models are not immune to cognitive biases—systematic deviations from rational judgment that can subtly influence their outputs, leading to skewed or suboptimal results. The paper 'Capturing Failures of Large Language Models via Human Cognitive Biases' discusses how machine learning systems, including advanced models like Codex, are affected by human cognitive biases in code generation. Our results are consistent with the findings of this paper, reflecting these biases' impact on model performance.

**How Does Our Fine-Tuned TinyLlama Model Compare to GPT-2 in Handling Cognitive Biases?**

After comparing our fine-tuned TinyLlama model (humaneval\_SFTTrainer\_model) with GPT-2 across 20 different prompts, including those addressing biases, and testing with over a hundred cases, we found that our model outperformed GPT-2. Using a pass@k evaluation, our model achieved an accuracy of 32%, while GPT-2 managed 21%. This demonstrates that our model handles various prompts more effectively and exhibits stronger overall performance, particularly in addressing complex scenarios.  
However, both models show weak results, largely because they were trained on text rather than on code like Codex. Additionally, as previous-generation models, their performance aligns with expectations based on the paper, 'Capturing Failures of Large Language Models via Human Cognitive Biases'. Despite the improvements, the significant reduction in biased influence highlights the need for further advancements in model training and evaluation.

**What Challenges Did We Face While Testing LLMs for Code Generation, and How Did humaneval\_SFTTrainer\_model and GPT-2 Perform in Code Generation Tests?**

Each test was performed on both models, often requiring minor adjustments, such as reformatting the prompts, as the models had difficulty generating outputs as functions rather than textual explanations. These models were primarily trained to work with text and therefore tend to output text rather than code. We needed them to perform code generation, so nearly every time we ran a test, we had to make some adjustments to guide the models correctly.  
This table contains measurements from approximately 20 tests conducted on the humaneval\_SFTTrainer\_model and GPT-2 model. It illustrates the behavior of each model, with consistent results observed across all tests, showing little variation from what is presented here.  
Each test took us about 30 minutes to perform on one model (we needed to use prompt engineering techniques to receive outputs from the models in the desired format). Therefore, conducting 20 tests took us approximately 20 hours to complete successfully. Additionally, there were numerous trials and errors along the way before we achieved acceptable results.

**Why Did We Opt for Previous Generation Models Over GPT-3.5 and GPT-4 for Code Generation?**

Newer models like GPT-3.5 have better performance in code generation; however, due to resource constraints (primarily computing), we opted to use models from the previous generation. We used ChatGPT-4 to generate prompts with biases, and we observed that both GPT-3.5, GPT-4, and Copilot Chat performed well and returned satisfactory results. However, we needed to gain experience with coding, and the previous generation models were readily available and could be run on local computers with minimal issues. Additionally, we attempted to use the GPT-4 model when we initially tried a custom Java-based GUI chatbox, but it required the use of a paid API.

**Why We Shifted from Java to Python for Our Work**

Initially, we attempted to use the GPT-4 model with a custom Java-based GUI chatbox, which we presented during our final class presentation. However, we encountered the issue of requiring a paid API, which led us to abandon this approach. Instead, we decided to run the prompts within the code base. Since the open-source models we used are implemented in Python and are available on platforms like HuggingFace and GitHub, we transitioned to working in Python. We utilized Amazon SageMaker Studio Lab , Google Colab , and Visual Studio  (limited by our own computer resources) for our experiments.

**Goals**

We successfully reached our goals: to understand the susceptibility of Large Language Models (LLMs) to different cognitive biases, such as gender bias, and to assess the impact of these biases on code generation tasks. One of our key strategies for mitigating these biases was to present the model with the simplest, easiest tasks, reducing the likelihood of mistakes and biased outputs.

**Difficulties and Interesting Findings**

Throughout the project, we faced several challenges. One significant difficulty was managing library dependencies, as different models required specific libraries, leading to potential conflicts. Additionally, our limited computing resources posed a challenge. While powerful services like Amazon SageMaker and Google Colab are available, they have limitations, such as usage time restrictions (4 hours) and long waiting times for GPU access due to server overloads. These issues led to delays in training and testing our models.

Despite these challenges, our findings were encouraging. Our custom model performed on par with or even outperformed the original prompts, particularly in reducing biases like gender bias and the framing effect. In most cases, our model significantly surpassed the biased prompts, highlighting its effectiveness in minimizing cognitive biases.

**Final Conclusion**

Our model, *finegptproject/humaneval\_SFTTrainer\_model*, trained on the HumanEval dataset, is not yet perfect in its understanding of natural language. However, our testing showed that it outperformed GPT-2, challenging the perspective we held after weeks of training and refinement.

In response to our research question—"To what extent does a custom GPT model enhance the accuracy, functionality, and bias reduction of AI-generated outputs when compared to existing AI tools, using prompts refined by the custom model and evaluated against the HumanEval dataset?"—we concluded that training LLMs to reduce biases can indeed lead to meaningful improvements. Our model achieved above 10% increase in accuracy, functionality, and bias reduction when evaluated against the GPT-2 on HumanEval dataset, particularly when using refined prompts and metrics like pass@k to evaluate performance.