CS598JBR MP Progress Report: Team-27

Maverick Espinosa, Muskan Shivnani, Sujay Nitish, Pranav Nagarajan
University of Illinois Urbana-Champaign
[mae10,mns9,snh6,pranavn6]@illinois.edu

1 Team Information

- GitHub Repository: https://github.com/emaverick2001/CS598JBR-Team-27
- Google Colab Workspace: https://colab.research.google.com/drive/1LuCFRL3Mkc-aPGimBkXpgLDiNAzLQqYM?usp=sharing

2 MP1

2.1 Pass@k Metric

The **pass@k** metric measures the probability that *at least one* of k generated solutions for a problem passes all unit tests.

Formally, let n be the total number of generated samples for a problem, k the number of draws/attempts without replacement (with $k \leq n$), and c the number of correct samples among the n. The per-problem pass@k is

$$\text{pass@}k \;=\; 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}} \qquad (k \leq n).$$

In practice, the reported dataset-level pass@k is the expectation (empirical mean) of this per-problem quantity across the dataset:

$$\text{pass@}k \;=\; \mathbb{E}_{\text{problems}}\left[1 - \frac{\binom{n-C}{k}}{\binom{n}{k}}\right],$$

which is commonly estimated by the sample average over M problems:

$$\widehat{\text{pass@}k} \;=\; \frac{1}{M}\sum_{i=1}^{M}\left(1 - \frac{\binom{n-c_i}{k}}{\binom{n}{k}}\right),$$

where c_i is the number of correct samples for problem i.

Notes / special cases.

- If $k = 1$, the formula reduces to pass@1 $= c/n$.
- If $n = 1$ (one generated sample per problem), then pass@1 equals the fraction of problems solved ("first-attempt" accuracy).
- When $k \ll n$, a useful approximation is pass@$k \approx 1 - \big((n-c)/n\big)^k$, which treats draws as approximately independent.

2.2 Pass@1 Results (Post-Processing)

The evaluation results after post-processing are:

- Base model (raw): 0.20 (4/20 problems solved)
- Base model (processed): 0.40 (8/20 problems solved)
- Instruct model (raw): 0.05 (1/20 problems solved)
- Instruct model (processed): 0.35 (7/20 problems solved)

2.3 Comparison Table

Problem_ID	Base	Base_Processed	Instruct	Instruct_Processed
HumanEval/109	Fail	Fail	Fail	Fail
HumanEval/112	Fail	Fail	Fail	Pass
HumanEval/119	Fail	Fail	Fail	Fail
HumanEval/134	Fail	Fail	Fail	Fail
HumanEval/142	Fail	Fail	Fail	Fail
HumanEval/143	Fail	Fail	Fail	Fail
HumanEval/145	Fail	Fail	Fail	Fail
HumanEval/147	Fail	Fail	Fail	Fail
HumanEval/152	Pass	Pass	Fail	Fail
HumanEval/3	Pass	Pass	Fail	Pass
HumanEval/36	Fail	Pass	Fail	Pass
HumanEval/39	Pass	Pass	Fail	Pass
HumanEval/4	Pass	Pass	Fail	Fail
HumanEval/64	Fail	Pass	Fail	Pass
HumanEval/76	Fail	Pass	Fail	Pass
HumanEval/78	Fail	Fail	Fail	Fail
HumanEval/84	Fail	Fail	Fail	Fail
HumanEval/92	Fail	Fail	Fail	Fail
HumanEval/95	Fail	Fail	Fail	Fail
HumanEval/97	Fail	Pass	Pass	Pass
Totals	4/20	8/20	1/20	7/20

Table 1: Pairwise comparison of results across Base and Instruct models (raw vs. processed).

2.4 Analysis

We highlight several key findings:

(1) **Base model improved after processing.** The number of solved problems doubled (from 4 to 8). This indicates that post-processing successfully removed extra functions, trailing code, and syntax noise.
(2) **Instruct model improved substantially.** The instruct model rose from only 1 correct problem to 7. Post-processing was particularly effective in extracting the intended main function from otherwise cluttered responses.
(3) **Patterns of fixes.** The main issues addressed were:
    - Removing duplicated or partial function definitions.
    - Stripping out embedded test functions and print statements.
    - Cleaning unfinished or unterminated docstrings.
(4) **Persistent failures.** Some problems (e.g., HumanEval/119, 142, 145) continued to fail due to incorrect logic in the generated code. These cannot be fixed by formatting-based post-processing alone.
(5) **Conclusion.** Both models meet the rubric's requirement: post-processing improved pass@1 by at least 20%. This demonstrates

that our pipeline is functioning correctly and robustly handles formatting issues in model outputs.

3 MP2

TBA

4 MP3

TBA