Critique Generation with Language Models for Code Generation and Arithmetic Reasoning Tasks

Eric Pan[1,2], Afra Feyza Akyürek[2], Dr. Derry Wijaya[2]

Winchester High School, 80 Skillings Rd, Winchester, MA 01890[1]

Department of Computer Science, Boston University, 111 Cummington Mall, Boston, MA 02215[2]

Introduction

In the field of natural language processing, artificially intelligent large language models (LLMs) are becoming more accurate and versatile than ever before, a trend accelerated by the introduction of **OpenAI's GPT-3** language prediction model. Using **few-shot learning**, GPT-3 is capable of adapting to diverse NLP tasks based on only a limited number of examples. One GPT-3 descendant, **OpenAI Codex**, applies its modeling capabilities to **code generation** and **math problem solving**, powering assistive tools like GitHub Copilot.[1]

Model	Released	Parameters
GPT-2	**Feb 2019**	**1.5 Billion**
GPT-3	**Jun 2020**	**175 Billion**

In this work, we examine Codex's ability not only to make correct predictions for the two tasks of code generation and math problem solving, but to provide **helpful critiques of incorrect solutions**. We accomplish this by designing additional prompts for critique generation. Through two experiments, we evaluate Codex's potential to **assist human learners**—providing insightful feedback into student solutions, possibly aiding classroom learning—and to **self-refine**, achieving **higher accuracy** by internalizing its own feedback.

Methods

Model: **OpenAI Codex** (code-davinci-002)
Benchmarks: **MBPP**[2] (dataset of Python coding tasks with unit tests), **GSM8K**[3] (dataset of math word problems with correct solutions)

Experiment 1: Assisting Humans

- We sampled **100 random problems** for each benchmark, then deliberately wrote an **erroneous prediction** for each problem.
- We sampled **4 critiques** for each of these incorrect solutions at temperatures 0.0, 0.2, 0.5, and 0.7.
- We evaluated how well each generated critique addressed the error, manually assigning **1 of 4 labels**. To ensure objectivity, we had multiple labelers compare their results, finding a **Jaccard similarity** that surpassed 0.80.

Experiment 2: Self-Refinement

- We tested Codex's **pass@1 accuracy*** on the MBPP sanitized split (427) and a random GSM8K test subset (400).

Baseline	Our Method
Generate	Generate + Self-Critique
Generate one solution for each problem. Then, assess accuracy of solutions.**	**Generate one solution for each problem. Then, generate one critique for each solution.** If a critique is negative, resample solution. Repeat resampling process for *n* iterations.**

*Percent of solutions that pass unit tests (MBPP) or result in correct answer (GSM8K) on the first evaluation
**All temperatures set to 0.7

Prompts

Generating Predictions

```
Problem #1: Write a Python
function to find the number of
divisors of a given integer.

Program: [BEGIN]
def num_divisors(n):
  divs = [i for i in range(1,
n+1) if n % i == 0]
  return len(divs)
[END]

[Completed problem sets #2-7]

Problem #8: Write a Python
function to check if an array is
monotonic.

Program: [Codex completes]
```

MBPP problem-solution example

Generating Critiques

```
[Completed problem sets #1-7]

Problem #8: There are 25 roses in
a garden. There are 40 tulips.
There are 35 daisies. What
percentage of flowers are not
roses?

Solution: There are 25+40+35=100
flowers total. There are 40+35=75
flowers that are not roses.
Therefore, (75/100)*100=75% of
the flowers are not roses.

Correct (y/n)? [Codex completes,
negative or affirmative]

Feedback: [Codex completes]
```

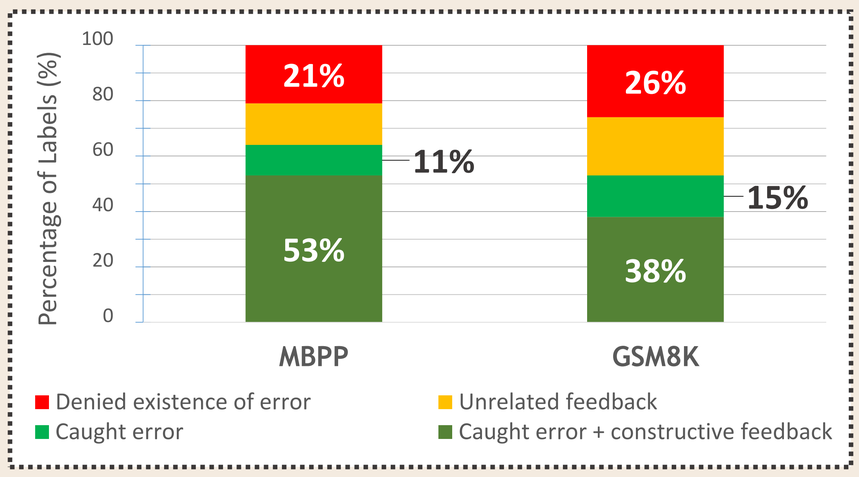
GSM8K problem-solution example

Few-shot learning models make predictions based on S training samples.[4] When prompting Codex, we limit $S = 7$.

Results

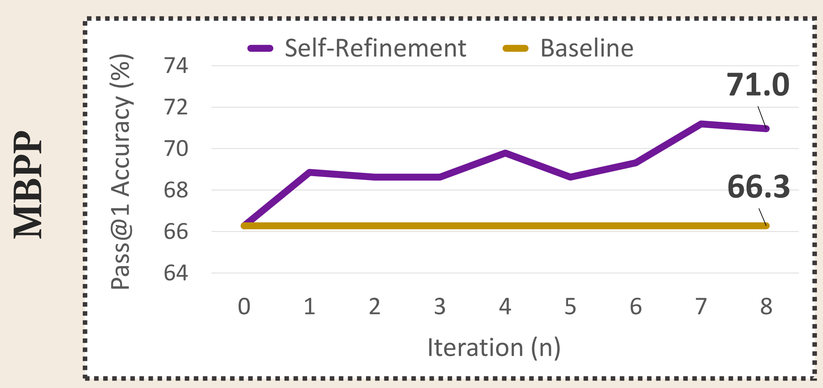
Experiment 1: Assisting Humans

We determined critique label distributions at temperatures 0.0, 0.2, 0.5, and 0.7. The chart below depicts distributions at optimal temperatures ($T_{MBPP} = 0.2$; $T_{GSM8K} = 0.0$).



Experiment 2: Self-Refinement

Dataset	Pass@1 (n = 0)	Pass@1 (n = 8)	Relative Improv.
MBPP	**66.3%**	**71.0%**	**+7.1%**
GSM8K	**55.8%**	**60.8%**	**+9.0%**



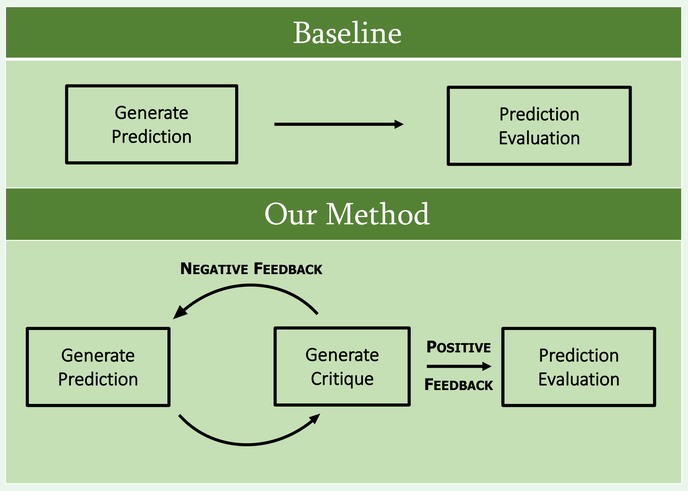
Discussion

While we conducted experiments using a specific model/dataset, **our proposed method of self-refinement** with **critique generation** is a generalized strategy **applicable to other NLP tasks** and requiring no fine-tuning.

Experiment 1: Assisting Humans

- Though more helpful than not, Codex's tendency to ignore errors suggests that it can **augment**—**but not replace**—teacher feedback.
- We recommend a **critique-selecting mechanism** (generating multiple critiques, then choosing the top *k* with a separate prompt) as a feature improving feedback quality in future work.

Experiment 2: Self-Refinement



- While **self-refinement** by itself leads to minor improvements in accuracy, it can be used in conjunction with other schemes.
- Effectiveness is limited by the **precision of the critique generator** in **giving negative or affirmative feedback**:

$$A_n = x_n \cdot P + (1 - x_n) \cdot A_o$$

A_n = pass@1 accuracy, iteration n A_0 = pass@1 accuracy, baseline
x_n = fraction of predictions given positive feedback, iteration n

References

[1] Chen, Mark, et al. "Evaluating Large Language Models Trained on Code." 2021.
[2] Austin, Jacob, et al. "Program Synthesis with Large Language Models." 2021.
[3] Cobbe, Karl, et al. "Training Verifiers to Solve Math Word Problems." 2021.
[4] Brown, Tom B., et al. "Language Models are Few-Shot Learners." 2020.

Acknowledgements

I am grateful for the patience, dedication, and counsel of my RISE research mentors, **Afra Feyza Akyürek** and **Dr. Derry Wijaya**. I would also like to thank the BU RISE coordinators for organizing this six-week long program.

BOSTON UNIVERSITY

LLAMA Language Learning Machine