J Statistical Confidence Intervals

	GPT-3.5		ChatGPT		GPT-4	
Task	Base	+SELF-REFINE	Base	+SELF-REFINE	Base	+SELF-REFINE
Sentiment Reversal	8.8 $\pm$ 2.05	**30.4** $\pm$ 3.61*	11.4 $\pm$ 2.34	**43.2** $\pm$ 3.98*	3.8 $\pm$ 1.28	**36.2** $\pm$ 3.82*
Dialogue Response	36.4 $\pm$ 6.14	**63.6** $\pm$ 6.62*	40.1 $\pm$ 6.33	**59.9** $\pm$ 6.67*	25.4 $\pm$ 5.36	**74.6** $\pm$ 6.22*
Code Optimization	14.8 $\pm$ 2.66	**23.0** $\pm$ 3.25*	23.9 $\pm$ 3.30	**27.5** $\pm$ 3.49	27.3 $\pm$ 3.48	**36.0** $\pm$ 3.81*
Code Readability	37.4 $\pm$ 6.86	**51.3** $\pm$ 7.39	27.7 $\pm$ 6.13	**63.1** $\pm$ 7.40*	27.4 $\pm$ 6.10	**56.2** $\pm$ 7.45*
Math Reasoning	**64.1** $\pm$ 3.47	**64.1** $\pm$ 3.47	74.8 $\pm$ 3.20	**75.0** $\pm$ 3.20	92.9 $\pm$ 2.05	**93.1** $\pm$ 2.03
Acronym Gen.	41.6 $\pm$ 7.72	**56.4** $\pm$ 8.15	27.2 $\pm$ 6.60	**37.2** $\pm$ 7.46	30.4 $\pm$ 6.92	**56.0** $\pm$ 8.15*
Constrained Gen.	28.0 $\pm$ 7.38	**37.0** $\pm$ 8.26	44.0 $\pm$ 8.72	**67.0** $\pm$ 9.00*	15.0 $\pm$ 5.38	**45.0** $\pm$ 8.77*

Table 13: SELF-REFINE results from table 1 with Wilson confidence interval (at 95% confidence interval) and statistical significance. On various tasks using GPT-3.5, ChatGPT, and GPT-4 as base LLM, SELF-REFINE consistently improves LLM. Metrics used for these tasks are defined in Section 3.2 as follows: Math Reasoning uses the solve rate; Code Optimization uses the percentage of programs optimized; and Sentiment Reversal, Dialogue Response and Acronym Gen use a GPT-4-based preference evaluation, which measures the percentage of times outputs from the base or enhanced models were selected, with the rest categorized as a tie. Constrained Gen uses the coverage percentage. Gains over Base, that are statistically significant based on these confidence intervals are marked *

Table 13 shows results from Table 1 with Wilson confidence interval (Brown et al., 2001) (at α= 99% confidence interval) and statistical significance. Gains that are statistical significance based on these confidence intervals are marked with an asterisk. We find that nearly all of GPT-4 gains are statistically significant, ChatGPT gains are significant for 4 out of 7 datasets, and GPT-3.5 gains are significant for 3 out of 7 datasets.

K New Tasks

Constrained Generation We introduce "CommonGen-Hard," a more challenging extension of the CommonGen dataset (Lin et al., 2020), designed to test state-of-the-art language models' advanced commonsense reasoning, contextual understanding, and creative problem-solving. CommonGen-Hard requires models to generate coherent sentences incorporating 20-30 concepts, rather than only the 3-5 related concepts given in CommonGen. SELF-REFINE focuses on iterative creation with introspective feedback, making it suitable for evaluating the effectiveness of language models on the CommonGen-Hard task.

Acronym Generation Acronym generation requires an iterative refinement process to create concise and memorable representations of complex terms or phrases, involving tradeoffs between length, ease of pronunciation, and relevance, and thus serves as a natural testbed for our approach. We source a dataset of 250 acronyms[4] and manually prune it to remove offensive or uninformative acronyms.

L Code Readability

Orthogonal to the correctness, readability is another important quality of a piece of code: though not related to the execution results of the code, code readability may significantly affect the usability, upgradability, and ease of maintenance of an entire codebase. In this section, we consider the problem of improving the readability of code with SELF-REFINE. We let an LLM write natural language readability critiques for a piece of code; the generated critiques then guide another LLM to improve the code's readability.

L.1 Method

Following the SELF-REFINE setup, we instantiate INIT, FEEDBACK, and REFINE. The INIT is a no-op — we directly start by critiquing the code with FEEDBACK and applying the changes with REFINE.

- **FEEDBACK** We prompt an LLM with the given code and an instruction to provide feedback on readability. We give the LLM the freedom to freely choose the type of enhancements and express them in the form of free text.
- **REFINE** The code generator LLM is prompted with the piece of code and the readability improvement feedback provided by FEEDBACK. In addition, we also supply an instruction to fix the code using the feedback. We take the generation from the code generator as the product of one iteration in the feedback loop.

Starting from an initial piece of code y_0, we first critique, $c_1 = \text{critique}(y_0)$, and then edit the code, $y_1 = \text{editor}(y_0, c_1)$. This is recursively performed N times, where $c_{k+1} = \text{critique}(y_k)$ and $y_{k+1} = \text{editor}(y_k, c_{k+1})$.

L.2 Experiments

Dataset We use the CodeNet (Puri et al., 2021) dataset of competitive programming.[5] For our purpose, these are hard-to-read multi-line code snippets. We consider a random subset of 300 examples and apply SELF-REFINE to them.

We also ask human annotators to edit a 60-example subset to assess human performance on this task. The human annotators are asked to read the code piece and improve its readability.

Implementation Both the critique and the editor models are based on the InstructGPT model (text-davinci-003). We consider the temperature of both $T = 0.0$ (greedy) and $T = 0.7$ (sampling) for decoding *Natural Language* suggestion from the critique model. We always use a temperature $T = 0.0$ (greedy) when decoding *Programming Language* from the code editor. Due to budget constraints, we run SELF-REFINE for $N = 5$ iterations. The exact prompts we use can be found in Figures 22-23.

[4] `https://github.com/krishnakt031990/Crawl-Wiki-For-Acronyms/blob/master/AcronymsFile.csv`

[5] `https://github.com/IBM/Project_CodeNet`

	Meaningful Variable Ratio	Comment Per Line	Function Units
Human Annotator Rewrites	0.653	0.24	0.70
SELF-REFINE (T = 0.0)	0.628	0.12	**1.41**
SELF-REFINE (T = 0.7)	**0.700**	**0.25**	1.33

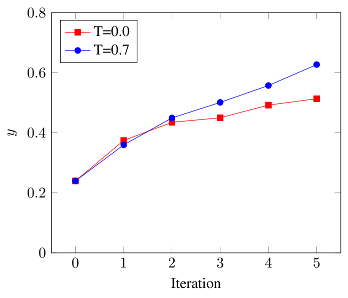
Table 14: Human v.s. SELF-REFINE performance on 60-example subset. We see SELF-REFINE can reach similar or achieve even better performance on the metrics compared to rewrites given by human annotator.

Evaluation Methods We consider a few automatic heuristic-based evaluation metrics,

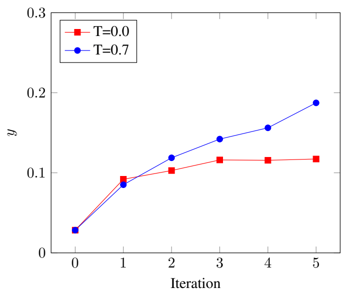
- Meaningful Variable Names: In order to understand the flow of a program, having semantically meaningful variable names can offer much useful information. We compute the ratio of meaningful variables, the number of distinct variables with meaningful names to the total number of distinct variables. We automate the process of extracting distinct variables and the meaningful subset of variables using a few-shot prompted language model.
- Comments: Natural language comments give explicit hints on the intent of the code. We compute the average number of comment pieces per code line.
- Function Units: Long functions are hard to parse. Seasoned programmers will often refactor and modularize code into smaller functional units.

Result For each automatic evaluation metric, the ratio of meaningful variable, of comment, and the number of function units, we compute for each iteration averaged across all test examples and plot for each SELF-REFINE iteration in Figure 11(a), Figure 11(b) and Figure 11(c) respectively. The two curves each correspond to critique with temperature $T = 0.0$ and $T = 0.7$. The iteration 0 number is measured from the original input code piece from CodeNet. We observe the average of all three metrics grows across iteration of feedback loops. A diverse generation of a higher temperature in the critique leads to more edits to improve the meaningfulness of variable names and to add comments. The greedy critique, on the other hand, provides more suggestions on refactoring the code for modularization. Figure 12 provides an example of code-readability improving over iterations.

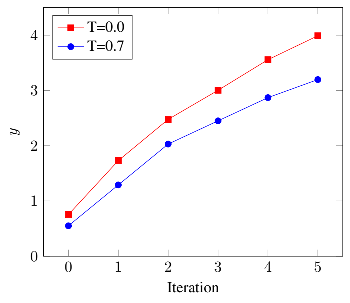
In Table 14, we measure human performance on all three metrics and compare with SELF-REFINE last iteration output. At $T = 0.7$, SELF-REFINE produces more meaning variables, more function units and slightly more comments compared to the human annotators on average. At $T = 0.0$, SELF-REFINE produces less meaningful variables, less comments per line but even more function units.



(a) Meaningful variable ratio across different SELF-REFINE iterations.



(b) Comment per line ratio across different SELF-REFINE iterations.



(c) Number of function units across different SELF-REFINE iterations.

Figure 11: Evaluation on code readability task with SELF-REFINE across multiple metrics

Example

M Dialogue Response Generation

Open-domain dialogue response generation is a complex task that requires a system to generate human-like responses to a wide range of topics. Due to the open-ended nature of the task, it is