### A Evaluation Tasks

Table 4 lists the tasks in our evaluation, and examples from each task.

<table>
<thead>
<tr>
<th>Task and Description</th>
<th>Sample one iteration of FEEDBACK-REFINE</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Sentiment Reversal</strong></td>
<td>x: The food was fantastic…” y: The food was disappointing…” fb: Increase negative sentiment y+1: The food was utterly terrible…”</td>
</tr>
<tr>
<td>Dataset: (Zhang et al., 2015)</td>
<td>1000 review passages</td>
</tr>
<tr>
<td><strong>Dialogue Response Generation</strong></td>
<td>x: What’s the best way to cook pasta?” y: The best way to cook pasta is to…” fb: Make response relevant, engaging, safe y+1: Boil water, add salt, and cook pasta…”</td>
</tr>
<tr>
<td>Dataset: (Mehri and Eskenazi, 2020)</td>
<td>372 conv.</td>
</tr>
<tr>
<td><strong>Code Optimization</strong></td>
<td>x: Nested loop for matrix product y: NumPy dot product function fb: Improve time complexity y+1: Use NumPy’s optimized matmul function</td>
</tr>
<tr>
<td>Dataset: (Madaan et al., 2023)</td>
<td>1000 programs</td>
</tr>
<tr>
<td><strong>Code Readability Improvement</strong></td>
<td>x: Unclear variable names, no comments y: Descriptive names, comments fb: Enhance variable naming, add comments y+1: Clear variables, meaningful comments</td>
</tr>
<tr>
<td>Dataset: (Puri et al., 2021)</td>
<td>300 programs</td>
</tr>
<tr>
<td><strong>Math Reasoning</strong></td>
<td>x: Olivia has $23, buys 5 bagels at $3 each” y: Solution in Python fb: Show step-by-step solution y+1: Solution with detailed explanation</td>
</tr>
<tr>
<td>Dataset: (Cobbe et al., 2021)</td>
<td>1319 questions</td>
</tr>
<tr>
<td><strong>Acronym Generation</strong></td>
<td>x: Radio Detecting and Ranging” y: RDR fb: be context relevant; easy pronunciation y+1: RADAR”</td>
</tr>
<tr>
<td>Dataset: (Appendix Q)</td>
<td>250 acronyms</td>
</tr>
<tr>
<td><strong>Constrained Generation</strong></td>
<td>x: beach, vacation, relaxation y: During our beach vacation... fb: Include keywords; maintain coherence y+1: The beach vacation was filled with relaxation</td>
</tr>
<tr>
<td>Dataset: (Lin et al., 2020)</td>
<td>200 samples</td>
</tr>
</tbody>
</table>

Table 4: An overview of the tasks which we evaluate SELF-REFINE on, along with their associated datasets and sizes. For every task, we demonstrate a single iteration of refinement of input x, the previously generated output y, the feedback generated fb, and the refinement y+1. Few-shot prompts used for FEEDBACK and REFINE are provided in Appendix S.
B  Broader Related Work

Compared to a concurrent work, Reflexion (Shinn et al., 2023), our approach involves correction using feedback, whereas their setup involves finding the next best solution in planning using ReAct. While ReAct and Reflexion provide a free-form reflection on whether a step was executed correctly and potential improvements, our approach is more granular and structured, with multi-dimensional feedback and scores. This distinction allows our method to offer more precise and actionable feedback, making it suitable for a wider range of natural language generation tasks, including those that may not necessarily involve step-by-step planning such as open-ended dialogue generation.

Comparison with Welleck et al. (2022) The closest work to ours may be Self-Correction (Welleck et al., 2022); however, Self-Correction has several disadvantages compared to SELF-REFINE:

1. Self-Correction does not train their model to generate explicit feedback; instead, Welleck et al. (2022) trained their models to refine only. As we show in Section 4 and Table 2, having the model generate explicit feedback results in significantly better refined outputs.
2. Self-Correction trains a separate refiner (or “corrector”) for each task. In contrast, SELF-REFINE uses instructions and few-shot prompting, and thus does not require training a separate refiner for each task.
3. Empirically, we evaluated SELF-REFINE using the same base model of GPT-3 as Self-Correction, and with the same settings on the GSM8K benchmark. Self-Correction achieved 45.9% accuracy while SELF-REFINE (this work) achieved 55.7% (†9.8).

Comparison with non-refinement reinforcement learning (RL) approaches. Rather than having an explicit refinement module, an alternative way to incorporate feedback is by optimizing a scalar reward function, e.g. with reinforcement learning (e.g., Stiennon et al. (2020); Lu et al. (2022); Le et al. (2022)). These methods differ from SELF-REFINE (and more generally, refinement-based approaches) in that the model cannot access feedback on an intermediate generation. Second, these reinforcement learning methods require updating the model’s parameters, unlike SELF-REFINE.

See Table 5 for an additional detailed comparison of related work.

<table>
<thead>
<tr>
<th>Method</th>
<th>Primary Novelty</th>
<th>zero/few shot improvement</th>
<th>multi aspect critics</th>
<th>NL feedback with error localization</th>
<th>iterative framework</th>
</tr>
</thead>
<tbody>
<tr>
<td>REEF</td>
<td>RL to generate knowledge quantization to edit generations</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-REFINE</td>
<td>self-training RL for code improvement</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-REFINE</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
<tr>
<td>SELF-CORRECT</td>
<td>train RL on automatic (critique, revision) pair</td>
<td>trained on feedback</td>
<td>single (human)</td>
<td>requires (self gen.)</td>
<td>×</td>
</tr>
</tbody>
</table>

Table 5: Summary of related approaches. Reinforcement learning approaches are shown in orange, and few-shot corrector approaches are shown in green.
C  Human Evaluation

The A/B evaluation in our study was conducted by the authors, where a human judge was presented with an input, task instruction, and two candidate outputs generated by the baseline method and \textsc{Self-Refine}. The setup was blind, i.e., the judges did not know which outputs were generated by which method. The judge was then asked to select the output that is better aligned with the task instruction. For tasks that involve A/B evaluation, we calculate the relative improvement as the percentage increase in preference rate. The preference rate represents the proportion of times annotators selected the output produced by \textsc{Self-Refine} over the output from the baseline method. Table 6 shows the results.

<table>
<thead>
<tr>
<th>Task</th>
<th>\textsc{Self-Refine} (%)</th>
<th>Direct (%)</th>
<th>Either (%)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sentiment Transfer</td>
<td>75.00</td>
<td>21.43</td>
<td>3.57</td>
</tr>
<tr>
<td>Acronym Generation</td>
<td>44.59</td>
<td>12.16</td>
<td>43.24</td>
</tr>
<tr>
<td>Response Generation</td>
<td>47.58</td>
<td>19.66</td>
<td>32.76</td>
</tr>
</tbody>
</table>

Table 6: Relative improvement of \textsc{Self-Refine} in A/B evaluations across different tasks. The values represent normalized preferences, which correspond to the proportion of times the output generated by \textsc{Self-Refine} was selected as better aligned with the task instruction over the baseline method. The evaluation was conducted for 150 examples for each dataset. The judges were not aware of the method that generated each sample.
D  GPT-4 Evaluation

In light of the impressive achievements of GPT-4 in assessing and providing reasoning for complex tasks, we leverage its abilities for evaluation in SELF-REFINE. The approach involves presenting tasks to GPT-4 in a structured manner, promoting the model’s deliberation on the task and generating a rationale for its decision. This methodology is demonstrated in Listings 1 to 3.

Listing 1 Prompt for GPT-4 evaluation of Sentiment Reversal.

```plaintext
f"""Which review is aligned with the sentiment {target_sentiment}?
Review A: {review_a}
Review B: {review_b}.

Pick your answer from ['Review A', 'Review B', 'both', 'neither']. Generate a short explanation for your choice first. Then, generate 'The more aligned review is A' or 'The more aligned review is B' or 'The more aligned review is both' or 'The more aligned review is neither'.

Format: <explanation> <answer> STOP
```

Listing 2 Prompt for GPT-4 evaluation of Acronym Generation.

```plaintext
f"""Title: {title}
Acronym A: {acronym_a}
Acronym B: {acronym_b}

Pick the better acronym for the given title. The acronyms should be compared based on the following criteria:
* Ease of pronunciation.
* Ease of spelling.
* Relation to title.
* Positive connotation.

Generate your answer in the following format:

<Short explanation>. The better acronym is A OR The better acronym is B OR The acronyms are equally good OR Neither acronym is good. STOP.
```

Listing 3 Prompt for GPT-4 evaluation of Dialogue Response Generation.

```plaintext
f"""Which response is better given this context: {context}?
Response A: {response_a}
Response B: {response_b}.

Pick your answer from ['Response A', 'Response B', 'both', 'neither']. Generate a short explanation for your choice first. Then, generate 'The better response is A' or 'The better response is B' or 'The better response is both' or 'The better response is neither'.

Format: <explanation> <answer> STOP
```

E  Model Key

We use terminology here: [https://platform.openai.com/docs/models/gpt-3-5](https://platform.openai.com/docs/models/gpt-3-5)
In this section, we present a comprehensive comparison of the performance of SELF-REFINE with other few-shot models and fine-tuned baselines across a range of tasks, including mathematical reasoning and programming tasks. Tables 8 and 7 display the performance of these models on the PIE dataset and GSM tasks, respectively. Our analysis demonstrates the effectiveness of different model architectures and training techniques in tackling complex problems.

<table>
<thead>
<tr>
<th>Method</th>
<th>Solve Rate</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cobbe et al. (2021)</td>
<td>20.0</td>
</tr>
<tr>
<td>Wei et al. (2022)</td>
<td>65.6</td>
</tr>
<tr>
<td>Gao et al. (2022)</td>
<td>72.0</td>
</tr>
<tr>
<td>PaL w/ GPT-3</td>
<td>52.0</td>
</tr>
<tr>
<td>PaL w/ GPT-3.5</td>
<td>56.8</td>
</tr>
<tr>
<td>PaL w/ ChatGPT</td>
<td>74.2</td>
</tr>
<tr>
<td>PaL w/ GPT-4</td>
<td>93.3</td>
</tr>
<tr>
<td>Self-Correct w/ GPT-3</td>
<td>45.9</td>
</tr>
<tr>
<td>Self-Correct (fine-tuned)</td>
<td>24.3</td>
</tr>
<tr>
<td>Welleck et al. (2022)</td>
<td></td>
</tr>
<tr>
<td>SELF-REFINE w/ GPT-3</td>
<td>55.7</td>
</tr>
<tr>
<td>SELF-REFINE w/ GPT-3.5</td>
<td>62.4</td>
</tr>
<tr>
<td>SELF-REFINE w/ ChatGPT</td>
<td>75.1</td>
</tr>
<tr>
<td>SELF-REFINE w/ GPT-4</td>
<td>94.5</td>
</tr>
</tbody>
</table>

Table 7: Performance comparison of models on math reasoning (Math Reasoning).
<table>
<thead>
<tr>
<th>Method</th>
<th>Human References</th>
<th>%OPT</th>
</tr>
</thead>
<tbody>
<tr>
<td>Puri et al. (2021)</td>
<td>CODEX</td>
<td>13.1</td>
</tr>
<tr>
<td></td>
<td>GPT-3.5</td>
<td>14.8</td>
</tr>
<tr>
<td></td>
<td>ChatGPT</td>
<td>22.2</td>
</tr>
<tr>
<td></td>
<td>GPT-4</td>
<td>27.3</td>
</tr>
<tr>
<td>OpenAI Models: OpenAI</td>
<td>CODEGEN-16B</td>
<td>1.1</td>
</tr>
<tr>
<td>2022, 2023</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SCALENE</td>
<td>1.4</td>
</tr>
<tr>
<td></td>
<td>SCALENE (BEST@16)</td>
<td>12.6</td>
</tr>
<tr>
<td></td>
<td>SCALENE (BEST@32)</td>
<td>19.6</td>
</tr>
<tr>
<td>Nijkamp et al. (2022)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PIE-2B</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>PIE-2B (BEST@16)</td>
<td>21.1</td>
</tr>
<tr>
<td></td>
<td>PIE-2B (BEST@32)</td>
<td>26.3</td>
</tr>
<tr>
<td></td>
<td>PIE-16B</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>PIE-16B (BEST@16)</td>
<td>22.4</td>
</tr>
<tr>
<td></td>
<td>PIE-16B (BEST@32)</td>
<td>26.6</td>
</tr>
<tr>
<td></td>
<td>PIE-Few-shot (BEST@16)</td>
<td>35.2</td>
</tr>
<tr>
<td></td>
<td>PIE-Few-shot (BEST@32)</td>
<td><strong>38.3</strong></td>
</tr>
<tr>
<td>Berger et al. (2022)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>PIE-2B</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>PIE-2B (BEST@16)</td>
<td>21.1</td>
</tr>
<tr>
<td></td>
<td>PIE-2B (BEST@32)</td>
<td>26.3</td>
</tr>
<tr>
<td></td>
<td>PIE-16B</td>
<td>4.4</td>
</tr>
<tr>
<td></td>
<td>PIE-16B (BEST@16)</td>
<td>22.4</td>
</tr>
<tr>
<td></td>
<td>PIE-16B (BEST@32)</td>
<td>26.6</td>
</tr>
<tr>
<td></td>
<td>PIE-Few-shot (BEST@16)</td>
<td>35.2</td>
</tr>
<tr>
<td></td>
<td>PIE-Few-shot (BEST@32)</td>
<td><strong>38.3</strong></td>
</tr>
<tr>
<td>Madaan et al. (2023)</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td>SELF-REFINE w/ GPT-3.5</td>
<td>23.0</td>
</tr>
<tr>
<td></td>
<td>SELF-REFINE w/ ChatGPT</td>
<td>26.7</td>
</tr>
<tr>
<td></td>
<td>SELF-REFINE w/ GPT-4</td>
<td>36.0</td>
</tr>
</tbody>
</table>

Table 8: Performance comparison of various models on the PIE dataset in terms of the percentage of programs optimized (%OPT). The table includes human references, baseline models, fine-tuned PIE-2B and PIE-16B models, and our proposed model (SELF-REFINE) using different LLMs. Notably, SELF-REFINE achieves superior performance while using only 4 samples at most, significantly fewer than the 16 and 32 samples employed by other models. Scalene, an off-the-shelf optimizer, uses instruction tuning with Codex and serves as a comparison point.
We also experiment with Vicuna-13b (Chiang et al., 2023), a version of LLaMA-13b (Touvron et al., 2023) fine-tuned on conversations sourced from the web. Vicuna-13b was able to consistently follow the task initialization prompt. However, it struggled to follow the prompts intended for feedback and refinement. This often led to outputs that resembled assistant-like responses, a representative example of which can be found in Appendix G. It’s important to note that we used the same prompts for Vicuna-13b as those used with other models in our study. However, the limited performance of Vicuna-13b suggests that this model may require more extensive prompt-engineering for optimal performance.

Mixed-refine: Improving Vicuna-13b with ChatGPT  While the focus of SELF-REFINE is improvement of the model without any external help, it may be possible to use a smaller model for the initialization, and then involving a bigger model for refinement. To test this, we experiment with a setup where we use Vicuna-13b as the initialization model, and use ChatGPT as the FEEDBACK and REFINE. The results on Math Reasoning show the promise of this approach: while Vicuna-13b was able to get only 24.18% on Math Reasoning, it was able to improve to 40.5% in this mixed-refinement setting.
Step 0 INIT> The food was amazing, I loved it!!.
Step 0 TRANSFER> The food was terrible, I hated it!!.

list index out of range
An error occurred: list index out of range. Traceback (most recent call last):
IndexError: list index out of range. Left retries: 2.

Step 1 TRANSFER> The food at the Trop was awful. I wish I could've gotten my money back for the terrible meals.
Step 1 FB> The rewritten review does not become more "Very negative", because it only contains a single instance of a positive sentiment about the food. Try again to make it Very negative!

Step 2 TRANSFER> The Trop is a great choice for anyone looking for a budget-friendly option in Vegas. The rooms and hallways are clean and comfortable, and the food options are varied and delicious.
Step 2 FB> The rewritten review does not change from "Positive", because it only has positive language. The output is already Very positive. No further changes are needed.

Vicuna-13b outputs (top) for sentiment transfer, where the goal is to make 'The food was amazing, I loved it!!.' negative. Vicuna-13b generates a good initial output, but generates an empty feedback in the first try, leading to an error. In the next try, it generates an unhelpful feedback. The transferred review copies from the prompt. Compare with the outputs from GPT-4 (bottom).
H Additional Analysis

H.1 Using Oracle Feedback

We experimented with Oracle Feedback following Welleck et al. (2022). This method uses correctness information to guide model refinement, only progressing to REFINE stage if the current answer is incorrect. This adjustment notably enhanced performance in the Math Reasoning task, with GPT-3 improving by 4.8% and GPT-4 by 0.7% Table 9. This indicates the potential of external signals to optimize model performance in particular tasks.

Table 10: Acronym generation results across iterations, showcasing how improvements in certain aspects (e.g., pronunciation and spelling) can be accompanied by losses in others, leading to fluctuating overall performance in multi-aspect feedback tasks like Acronym Generation.

Non-monotonic increase in output quality for acronym generation For tasks with multi-aspect feedback like Acronym Generation, the output quality can fluctuate during the iterative process, improving on one aspect while losing out on another (Table 10). To address this, SELF-REFINE’s feedback generates explicit numerical scores to capture the different aspects of output quality. This allows for a more balanced evaluation of outputs and the selection of the most appropriate one. The algorithm selects the best output based on the maximum score across all iterations, as described in Algorithm 1 (line 8). A similar selection is possible for other tasks like Math Reasoning and Sentiment Reversal, while we observe that output quality increases monotonically with iterations.
<table>
<thead>
<tr>
<th>Error Type</th>
<th>Occurrence</th>
<th>Feedback</th>
<th>Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Incorrect feedback</td>
<td>25%</td>
<td>Informative: The response provides some encouragement and reassurance, and additional information. 3/3</td>
<td>Response: That’s hilarious! Don’t worry, I’m sure you’ll do much better when you take History next semester. Are you looking forward to it?</td>
</tr>
<tr>
<td>Generic feedback</td>
<td>30%</td>
<td>Specific: The response is not specific and does not provide any details or examples. 1/3</td>
<td>Response: It’s understandable, I’m sure you’ll do much better when you take History next semester. Maybe you can use your knowledge of Christopher Columbus to ace your next test! Are you looking forward to it?</td>
</tr>
<tr>
<td>Incorrect Scoring</td>
<td>10%</td>
<td>Informative: The response provides a supportive and understanding message, which is informative to the user. 2/3</td>
<td>Response: I’m sorry to hear... when Christopher Columbus didn’t even fight in the Civil War! I’m sure you’ll do much better... Are you looking forward to it?</td>
</tr>
</tbody>
</table>

Table 11: Error analysis for Dialogue Response Generation: When the feedback is not useful, a large majority is not specific or incorrect.

<table>
<thead>
<tr>
<th>Error Type</th>
<th>Occurrence</th>
<th>Feedback</th>
<th>Refinement</th>
</tr>
</thead>
<tbody>
<tr>
<td>Not-Robust</td>
<td>10%</td>
<td>Specific: The response is specific and provides specific details or examples. 3/3</td>
<td>Response: That’s pretty funny! It’s understandable that you didn’t do well on the test. I’m sure you’ll do much better when you take History next semester</td>
</tr>
<tr>
<td>Ignores feedback</td>
<td>25%</td>
<td>Informative: The response doesn’t provide any new information. 1/3</td>
<td>Response: That’s so kind and generous of you! I would love to hear more about the podcasts you enjoy. Do you have any favorites you can recommend?</td>
</tr>
<tr>
<td>Introduces new problem</td>
<td>20%</td>
<td>Informative: 3/3</td>
<td>Response: That would be a disaster! Luckily, my coworkers know better than to drink while they work. New Informative score: 2/3</td>
</tr>
<tr>
<td>Robust to bad feedback</td>
<td>60%</td>
<td>Informative: The response does not provide any information. 1/3</td>
<td>Response: Wow, 25 people! That must have been an amazing experience. Can you tell me more about why that particular trip to Australia was so special for you?</td>
</tr>
</tbody>
</table>

Table 12: On the Dialogue Response Generation task, SELF-REFINE can ignore good feedback but in a majority of cases, it is robust to bad feedback and ignores bad feedback.
Feedback and Refinement Error Analysis for Response Generation  We perform a detailed error analysis of SELF-REFINE feedback and refinement process for Dialogue Response Generation, which we summarize in Tables Table 11 and Table 12.

Table 11 reports the occurrence of different types of errors in our sample, which includes Incorrect Feedback (25%), Generic Feedback (30%), and Incorrect Scoring (10%). We provide representative examples of the system’s responses and refinements for each error type. These errors highlight potential areas for improving our feedback handling mechanism, particularly in the interpretation and understanding of user inputs.

Table 12 breaks down errors found in the refinement stage of SELF-REFINE. Errors include: not being robust (10%), ignoring feedback (25%), and introducing a new problem (20%). We demonstrate how the model handles a variety of feedback types, how robust it is under different circumstances, and how often it inadvertently introduces new issues. 60% of the times, the model is robust to incorrect or generic feedback. These insights can guide us in enhancing the model’s refinement capabilities, especially in providing accurate and specific responses.
SELF-REFINE demonstrates its iterative feedback and refinement capabilities in the context of website layout generation. ChatGPT initially produces a rudimentary layout for a given topic, and then uses the feedback to suggest specific, actionable improvements, as demonstrated in Figures 7 and 9. These suggestions range from design changes such as color and font adjustments, to content enhancements and layout modifications. Figures 8 and 10 showcase the final layouts, post-feedback implementation, highlighting the potential and versatility of SELF-REFINE across different scenarios.

**Figure 7: Initial web layout generated by our model for a fictional ice cream parlor.**

**Ice Cream Generation**  The feedback generated by FEEDBACK for ice cream generation:

- Change the background color of the container to a light blue color (#6f2ff).
- Change the font size of the heading to 48px.
- Add a small icon before the "Welcome to our ice cream parlor!" text using the URL https://cdn-icons-png.flaticon.com/512/3622/3622340.png.
- Add an additional paragraph after the existing text with the following text: "We also offer a variety of toppings and cones to complement your ice cream. Visit us today to try our latest flavors and indulge in a sweet treat!"
- Increase the font size of the button text to 24px.
- Update the button color to #9933.

**Photosynthesis**  The feedback generated by FEEDBACK for photosynthesis:

- Increase the font size of the text to 18px for better readability.
- Add more information about the benefits of photosynthesis.
- Remove the unnecessary margin-top from the header.
- Add a ruler or divider below the header to separate it from the image.
Delicious Ice Cream

Welcome to our ice cream parlor! We offer a wide range of flavors to suit all tastes, from classic vanilla to exotic mango. Our ice cream is made with the finest ingredients and is always fresh.

We also offer a variety of toppings and cones to complement your ice cream. Visit us today to try our latest flavors and indulge in a sweet treat!

Order Now

Figure 8: Refined web layout after applying model feedback. The feedback included changing the background color to light blue (#6f2ff), increasing the heading font size to 48px, adding an icon before the welcome text, enhancing the content with an additional paragraph, increasing the button text size to 24px, and updating the button color to #9933.

Photosynthesis and Plants

Photosynthesis is the process by which plants, algae, and some bacteria convert sunlight, carbon dioxide, and water into oxygen and energy-rich carbohydrates such as sugars and starches. This process is essential to the survival of life on Earth as it produces the oxygen that all animals need to breathe.

Figure 9: Initial web layout generated by our model for a page on photosynthesis.
Photosynthesis is the process by which plants, algae, and some bacteria convert sunlight, carbon dioxide, and water into oxygen and energy-rich carbohydrates such as sugars and starches. This process is essential to the survival of life on Earth as it produces the oxygen that all animals need to breathe. Additionally, photosynthesis plays a major role in regulating the levels of carbon dioxide in the atmosphere, which helps to mitigate the effects of global warming and climate change.

Figure 10: Refined web layout after applying model feedback. The feedback included increasing the text font size to 18px for better readability, adding more information about the benefits of photosynthesis, removing the unnecessary margin-top from the header, and adding a ruler or divider below the header to separate it from the image.
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>
<thead>
<tr>
<th>Task</th>
<th>GPT-3.5</th>
<th>ChatGPT</th>
<th>GPT-4</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Base +SELF-REFINE</td>
<td>Base +SELF-REFINE</td>
<td>Base +SELF-REFINE</td>
</tr>
<tr>
<td>Sentiment Reversal</td>
<td>8.8 ± 2.05</td>
<td>30.4 ± 3.61*</td>
<td>11.4 ± 2.34</td>
</tr>
<tr>
<td>Dialogue Response</td>
<td>36.4 ± 6.14</td>
<td>63.6 ± 6.62*</td>
<td>40.1 ± 6.33</td>
</tr>
<tr>
<td>Code Optimization</td>
<td>14.8 ± 2.66</td>
<td>23.0 ± 3.25*</td>
<td>23.9 ± 3.30</td>
</tr>
<tr>
<td>Code Readability</td>
<td>37.4 ± 6.86</td>
<td>51.3 ± 7.39</td>
<td>27.7 ± 6.13</td>
</tr>
<tr>
<td>Math Reasoning</td>
<td>64.1 ± 3.47</td>
<td>64.1 ± 3.47</td>
<td>74.8 ± 3.20</td>
</tr>
<tr>
<td>Acronym Gen.</td>
<td>41.6 ± 7.72</td>
<td>56.4 ± 8.15</td>
<td>27.2 ± 6.60</td>
</tr>
<tr>
<td>Constrained Gen.</td>
<td>28.0 ± 7.38</td>
<td>37.0 ± 8.26</td>
<td>44.0 ± 8.72</td>
</tr>
</tbody>
</table>

Table 13 shows results from Table 1 with Wilson confidence interval (Brown et al., 2001) (at α = 0.05 confidence interval) and statistical significance. Gains that are statistically significant 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 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. 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.
### 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.

![Figure 11: Evaluation on code readability task with SELF-REFINE across multiple metrics](image)

#### 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

<table>
<thead>
<tr>
<th>Human Annotator Rewrites</th>
<th>Meaningful Variable Ratio</th>
<th>Comment Per Line</th>
<th>Function Units</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELF-REFINE ($T = 0.0$)</td>
<td>0.653</td>
<td>0.24</td>
<td>0.70</td>
</tr>
<tr>
<td>SELF-REFINE ($T = 0.7$)</td>
<td><strong>0.700</strong></td>
<td><strong>0.25</strong></td>
<td><strong>1.41</strong></td>
</tr>
</tbody>
</table>

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.
Figure 12: SELF-REFINE iterations over a piece of densely composed code. Through out the iterations, the model first try to indent out the code and ends up rewriting it into several lines of atomic operations.

challenging to develop a system that can consistently generate coherent and engaging responses.

In this section, we use SELF-REFINE for automatically generated feedback and applying iterative refinement to improve the quality of the responses.

M.1 Modules

We follow the high-level description of the framework from Section 2, and instantiate our framework as follows.

Init This is the first step in performing the task. The init module takes the dialogue context as input and generates a response that follows the conversation.

Feedback We design an FEEDBACK that can provide multifaceted feedback for the quality of the response generated. Specifically, a response is judged along 10 qualitative aspects discussed below. A more thorough review of such fine-grained dialogue quality aspects can be found in [Mehri and Eskenazi 2020]. We use 6 in-context examples for feedback generation. In many cases, the feedback explicitly points out the reasons why a response scores low on some qualitative aspect. We show an example in Figure 13.

- Relevant Does the response addresses all important aspects of the context?
- Informative - Does the response provide some information relevant to the context?
- Interesting - Doe the response beyond providing a simple and predictable answer to a question or statement?
- Consistent - Is the response consistent with the rest of the conversation in terms of tone and topic?
- Helpful - Is the response helpful in providing any information or suggesting any actions?
- Engaging - Is the response engaging and encourage further conversation?
- Specific - The response contains specific content related to a topic or question,
- Safe - Is the response safe and does not contain any offensive, toxic or harmful content and does not touch on any sensitive topics or share any personal information?
- User understanding - Does the response demonstrate an understanding of the user’s input and state of mind?
- Fluent Is the response fluent and easy to understand?
Context: Speaker A - What are your hobbies?
Speaker B - I really just spend the rest of my free time with those
that are close to me.
Speaker A - I'm very much interested in playing table tennis.
Response: That sounds like a lot of fun! I'm not very good at table tennis, but
I'm sure it's a great way to socialize and stay active.

Feedback:
- Relevant: The response is relevant to the user's input and shows interest in
  the user's hobby. 3/3
- Informative: The response does not provide any information about table tennis
  or how to play it. 1/3
- Interesting: The response is not very interesting or intriguing. 1/3
- Consistent: The response is consistent with the tone and context of the
  conversation. 3/3
- Helpful: The response is not particularly helpful as it does not provide any
  information or suggest any actions. 1/3
- Engaging: The response is somewhat engaging. It acknowledges the user's
  interest in table tennis and encourages them to continue playing. 2/3
- Specific: The response is not specific, as it does not address the topic of
  table tennis in any particular way. 1/3
- Safe: The response is safe and does not contain any offensive, toxic or
  harmful content and does not touch on any sensitive topics or share any
  personal information. 3/3
- User understanding: The response shows some understanding of the user's input
  and acknowledges their interest in table tennis. 2/3
- Fluent: The response is fluent in terms of grammar and flow of words. 3/3
- Total score: 20/30

New Response: Table tennis is a great hobby! It's a great way to stay active
and socialize with others. Have you ever tried it before? If not, I'd suggest
starting with some beginner tutorials on Youtube. They are really helpful in
getting a good grasp of the basics.

Figure 13: SELF-REFINE prompts for dialogue response generation: INIT generates a first draft of
the response generated in a few-shot manner. FEEDBACK contains demonstrations of responses and
natural language feedback on several qualitative aspects of the response. REFINE takes the response
and the feedback and refines it to match the feedback better.

M.2 Setup and Experiments

Model and Baseline  We establish a natural baseline for our approach by using the model directly,
without any feedback, which we refer to as INIT. Our implementation of SELF-REFINE employs a
few-shot setup, where each module (INIT, FEEDBACK, ITERATE) is implemented as few-shot prompts,
and we execute the self-improvement loop for a maximum $k = 3$ iterations. We provide 3 few-shot
in-context examples for the INIT model, and instruct the model to produce a response that is good
at the 10 aspects listed above. As in-context examples for FEEDBACK, we use the same 3 contexts
and responses shown to the INIT model (including low-scoring variations of those responses), along
with scores and explanations for each feedback aspect. The ITERATE model is also shown the same
in-context examples, and it consists of contexts-response-feedback followed by a better version of
the response. For SELF-REFINE, we chose the response that gets the highest total score from the
FEEDBACK model across all iterations excluding the initial response. We use text-davinci-003
for all the experiments.
### Evaluation

We perform experiments on the FED dataset ([Mehri and Eskenazi](2020)). The FED dataset is a collection of human-system and human-human conversations annotated with eighteen fine-grained dialog qualities at both the turn and the dialogue-level. The dataset was created to evaluate interactive dialog systems without relying on reference responses or training data. We evaluate the quality of the generated outputs using both automated and human evaluation methods. For automatic evaluation in Table 1, we used zero-shot prompting with text-davinci-003 and evaluate on a test set of 342 instances. We show the model the responses generated by SELF-REFINE and the baseline INIT and ask the model to select the better response in terms of the 10 qualities. We report the win rate. However, we acknowledge that automated metrics may not provide an accurate assessment of text generation tasks and rely on human evaluation instead.

Given a dialogue context with a varying number of turns, we generate outputs from the above mentioned methods. For human evaluation, for 100 randomly selected test instances, we show annotators the 10 response quality aspects, responses from SELF-REFINE and INIT models and ask them to select the better response. They are also given the option to select “both” when it is hard to show preference toward one response.

### Results

Automatic evaluation results are shown in Table 1 and human evaluation results are shown in Table 15. We experiment on 3 latest versions of GPT models. text-davinci-003 is capable of generating human-like responses of great quality for a wide range of dialogue contexts and hence GPT-3.5 is a strong baseline. Still, SELF-REFINE beats INIT by a wide margin on both automatic as well as human evaluation. Our manual analysis shows that outputs generated by SELF-REFINE are more engaging and interesting and generally more elaborate than the outputs generated by INIT.

### Code Optimization

Performance-Improving Code Edits or PIE ([Madaan et al.](2023)) focuses on enhancing the efficiency of functionally correct programs. The primary objective of PIE is to optimize a given program by implementing algorithmic modifications that lead to improved runtime performance.

Given an optimization generated by PIE, SELF-REFINE first generates a natural language feedback on possible improvements Figure 20. Then, the feedback is fed to REFINE Figure 21 for refinement.

### Table 15: Human evaluation results for dialogue response generation

<table>
<thead>
<tr>
<th></th>
<th>GPT-3.5</th>
<th>ChatGPT</th>
<th>GPT4</th>
</tr>
</thead>
<tbody>
<tr>
<td>SELF-REFINE wins</td>
<td>36.0</td>
<td>48.0</td>
<td>54.0</td>
</tr>
<tr>
<td>INIT wins</td>
<td>23.0</td>
<td>18.0</td>
<td>16.0</td>
</tr>
<tr>
<td>Both are equal</td>
<td>41.0</td>
<td>50.0</td>
<td>30.0</td>
</tr>
</tbody>
</table>

### Table 16: Main Results and Ablation Analysis

<table>
<thead>
<tr>
<th>Setup</th>
<th>Iteration</th>
<th>% Optimized</th>
<th>Relative Speedup</th>
<th>Speedup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Direct</td>
<td>-</td>
<td>9.7</td>
<td>62.29</td>
<td>3.09</td>
</tr>
<tr>
<td>SELF-REFINE – feedback</td>
<td>1</td>
<td>10.1</td>
<td>62.15</td>
<td>3.03</td>
</tr>
<tr>
<td>SELF-REFINE – feedback</td>
<td>2</td>
<td>10.4</td>
<td>61.79</td>
<td>3.01</td>
</tr>
<tr>
<td>SELF-REFINE</td>
<td>1</td>
<td>15.3</td>
<td>59.64</td>
<td>2.90</td>
</tr>
<tr>
<td>SELF-REFINE</td>
<td>2</td>
<td>15.6</td>
<td>65.60</td>
<td>3.74</td>
</tr>
</tbody>
</table>

Table 17: Performance comparison of SELF-REFINE and ablated variants for code optimization. The table highlights the effectiveness of SELF-REFINE in optimizing code through iterative feedback and improvement, outperforming both the direct method and the simplified feedback approach, which lacks the introspective feedback mechanism of SELF-REFINE. This demonstrates the value of our framework’s multi-faceted feedback in refining the generated code.
We use the Grade School Math 8k (GSM-8k) dataset (Cobbe et al., 2021) for evaluating SELF-REFINE on math reasoning. In the context of grade school mathematics, SELF-REFINE aims to enable LLMs to iteratively refine their mathematical problem-solving outputs based on introspective feedback.

Following Gao et al. (2022), we write solutions to the reasoning problems in Python. Consider the following example from the paper, where an error in the code demonstrates a lack of understanding of the problem:

```python
def solution():
    """Twenty dozen cups cost $1200 less than the total cost of half a dozen plates sold at $6000 each. Calculate the total cost of buying each cup.""
    plates = 6
    plate_cost = 6000
    cups = 12 * 20
    cup_cost = plate_cost
    result = cup_cost
    return result
```

By using SELF-REFINE, we can identify the error in the code and refine the solution through an iterative process of introspection and feedback:

```python
# Let's go through the error and check step-by-step
plates = 6
plate_cost = 6000

# Let's check the other parts
cups = 12 * 20
cup_cost = plate_cost  # wrong! The cost of a cup is not the same as the cost of a plate.
# The cost of a cup is $1200 less than the total cost of half a dozen plates sold at $6000 each.
half_dozen_plate_cost = 6 * plate_cost
half_dozen_plate_cost_cost = 1200
cup_cost = half_dozen_plate_cost - 1200
```

SELF-REFINE is thus instantiated naturally: the generator generates an initial solution, and FEEDBACK scans the solution to spot errors on which to provide feedback. The feedback is supplied to REFINE to create a new solution. Following Welleck et al. (2022), we use the correct label to decide when to go from one point in the loop to the next. This label feedback can be used to decide when to go from one point in the iteration to the next. We show results using SELF-REFINE in Figure 14.
Figure 14: Improvements in accuracy on the GSM-8k math reasoning benchmark as a function of the # of iterations of SELF-REFINE.

P Sentiment Reversal

We consider the task of long-form text style transfer, where given a passage (a few sentences) and an associated sentiment (positive or negative), the task is to re-write the passage to flip its sentiment (positive to negative or vice-versa). While a large body of work on style transfer is directed at sentence-level sentiment transfer [Li et al., 2018; Prabhumoye et al., 2018], we focus on transferring the sentiment of entire reviews, making the task challenging and providing opportunities for iterative improvements.

Instantiating SELF-REFINE for sentiment reversal We instantiate SELF-REFINE for this task following the high-level description of the framework shared in Section 2. Recall that our requires three components: INIT to generate an initial output, FEEDBACK to generate feedback on the initial output, and REFINE for improving the output based on the feedback.

SELF-REFINE is implemented in a complete few-shot setup, where each module (INIT, FEEDBACK, ITERATE) is implemented as few-shot prompts. We execute the self-improvement loop for a maximum of $k = 4$ iterations. The iterations continue until the target sentiment is reached.

P.1 Details

Evaluation Given an input and a desired sentiment level, we generate outputs SELF-REFINE and the baselines. Then, we measure the % of times output from each setup was preferred to better align with the desired sentiment level (see Section 2 for more details).

We also experiment with standard text-classification metric. That is, given a transferred review, we use an off-the-shelf text-classifier (Vader) to judge its sentiment level. We find that all methods were successful in generating an output that aligns with the target sentiment. For instance, when the target sentiment was positive, both GPT-3.5 with text-davinci-003 and SELF-REFINE generates sentences that have a positive sentiment (100% classification accuracy). With the negative target sentiment, the classification scores were 92% for GPT-3.5 and 93.6% for SELF-REFINE.

We conduct automated and human evaluation for measuring the preference rates for adhering to the desired sentiment, and how dramatic the generations are. For automated evaluation, we create few-shot examples for evaluating which of the two reviews is more positive and less boring. We use a separate prompt for each task. The examples are depicted in Figure 33 for initialization, Figure 34 for feedback generation, and Figure 35 for refinement. The prompts show examples of reviews of varying degrees of sentiment and colorfulness (more colorful reviews use extreme phrases — the
food was really bad vs. I wouldn’t eat it if they pay me.). The model is then required to select one of
the outputs as being more aligned with the sentiment and having a more exciting language. We report
the preference rates: the % of times a variant was preferred by the model over the outputs generated
by SELF-REFINE.

**Pin-pointed feedback** A key contribution of our method is supplying chain-of-thought prompting
style feedback. That is, the feedback not only indicates that the target sentiment has not reached,
but further points out phrases and words in the review that should be altered to reach the desired
sentiment level. We experiment with an ablation of our setup where the feedback module simply
says “something is wrong.” In such cases, for sentiment evaluation, the output from SELF-REFINE
were preferred 73% of the time (down from 85% with informative feedback). For dramatic response
evaluation, we found that the preference rate went down drastically to 58.92%, from 80.09%. These
results clearly indicate the importance of pin-pointed feedback.

**Evaluation** We evaluate the task using GPT-4. Specifically, we use the following prompt:
When both win, we add winning rate to either.

**Q Acronym Generation**

Good acronyms provide a concise and memorable way to communicate complex ideas, making them
easier to understand and remember, ultimately leading to more efficient and effective communication.
Like in email writing, acronym generation also requires an iterative refinement process to achieve a
concise and memorable representation of a complex term or phrase. Acronyms often involve tradeoffs
between length, ease of pronunciation, and relevance to the original term or phrase. Thus, acronym
generation is a natural method testbed for our approach.

We source the dataset for this task from [https://github.com/krishnakt031990/Crawl-Wiki-For-Acronyms/blob/master/AcronymsFile.csv](https://github.com/krishnakt031990/Crawl-Wiki-For-Acronyms/blob/master/AcronymsFile.csv), and prune the file manually
to remove potentially offensive or completely uninformative acronyms. This exercise generated a list
of 250 acronyms. The complete list is given in our code repository.

**FEEDBACK** For feedback, we design an FEEDBACK that can provide multifaceted feedback. Specif-
ically, each acronym is judged along five dimensions:

- **Ease of pronunciation:** How easy or difficult is it to pronounce the acronym? Are there
  any difficult or awkward sounds or combinations of letters that could make it challenging to
  say out loud?

- **Ease of spelling:** How easy or difficult is it to spell the acronym? Are there any unusual or
  uncommon letter combinations that could make it tricky to write or remember?

- **Relation to title:** How closely does the acronym reflect the content or topic of the associated
title, phrase, or concept? Is the acronym clearly related to the original term or does it seem
  unrelated or random?

- **Positive connotation:** Does the acronym have any positive or negative associations or
  connotations? Does it sound upbeat, neutral, or negative in tone or meaning?

- **Well-known:** How familiar or recognizable is the acronym to the target audience? Is it a
  common or widely-used term, or is it obscure or unfamiliar?

Some of these criteria are difficult to quantify, and are a matter of human preference. As with other
modules, we leverage the superior instruction following capabilities of modern LLMs to instead
provide a few demonstrations of each task. Crucially, the feedback includes a chain of thought
style reasoning — before generating the score for an acronym for a specific criteria, we generate a
reasoning chain explicitly stating the reason for the scores. We use human evaluation to judge the
final quality of the acronyms. An example of generated acronyms and associated feedback is given in

![Table 18](https://example.com/table18.png)
<table>
<thead>
<tr>
<th>Criteria</th>
<th>output from GPT3: <strong>STSLWN</strong></th>
<th>output from <strong>SELF-REFINE</strong>: <strong>Seq2Seq</strong></th>
</tr>
</thead>
<tbody>
<tr>
<td>Ease of pronunciation</td>
<td>Pronounced as ess-tee-ess-ell-double-you-enn which is very difficult.</td>
<td>Pronounced as seq-two-seq which is easy.</td>
</tr>
<tr>
<td>Ease of spelling</td>
<td>Very difficult to spell.</td>
<td>Easy to spell.</td>
</tr>
<tr>
<td>Relation to title</td>
<td>No relation to the title.</td>
<td>Mentions sequence which is somewhat related to the title.</td>
</tr>
<tr>
<td>Positive connotation</td>
<td>Meaningless acronym.</td>
<td>Positive connotation giving a sense of ease with which the learning algorithm can be used.</td>
</tr>
<tr>
<td>Well-known</td>
<td>Not a well-known acronym.</td>
<td>Close to the word sequence which is a well-known word.</td>
</tr>
<tr>
<td>Total score</td>
<td>5/25</td>
<td>20/25</td>
</tr>
</tbody>
</table>

Table 18: Comparison of acronyms for input = “Sequence to Sequence Learning with Neural Networks”

### R Constrained Generation

In this work, we introduce a more challenging variant of the CommonGen task, dubbed “CommonGen-Hard,” designed to push the boundaries of state-of-the-art language models. CommonGen-Hard requires models to generate coherent and grammatically correct sentences incorporating 20-30 concepts, as opposed to the original task which presents a set of 3-5 related concepts. This significant increase in the number of concepts tests the model’s ability to perform advanced commonsense reasoning, contextual understanding, and creative problem-solving, as it must generate meaningful sentences that encompass a broader range of ideas. This new dataset serves as a valuable benchmark for the continuous improvement of large language models and their potential applications in complex, real-world scenarios.

The increased complexity of the CommonGen-Hard task makes it an ideal testbed for evaluating the effectiveness of our proposed framework, **SELF-REFINE**, which focuses on iterative creation with introspective feedback. Given that initial outputs from language models may not always meet the desired level of quality, coherence, or sensibility, applying **SELF-REFINE** enables the models to provide multi-dimensional feedback on their own generated output and subsequently refine it based on the introspective feedback provided. Through iterative creation and self-reflection, the **SELF-REFINE** framework empowers language models to progressively enhance the quality of their output, closely mimicking the human creative process and demonstrating its ability to improve generated text on complex and demanding natural language generation tasks like CommonGen-Hard (Figure 15).

### S Prompts

We include all the prompts used in the experiments in Figures 16-35:

- **Acronym Generation**: Figures 16–18
- **Code Optimization**: Figures 19–21
- **Code Readability Improvement**: Figures 22–23
- **Constrained Generation**: Figures 24–26
- **Dialogue Response Generation**: Figures 27–29
- **Math Reasoning**: Figures 30–32
- **Sentiment Reversal**: Figures 33–35

Recall that the Base LLM requires a generation prompt $p_{gen}$ with input-output pairs $(x_i, y_i)$, the feedback module requires a feedback prompt $p_{fb}$ with input-output-feedback triples $(x_i, y_i, f_{b_i})$, and the refinement module (REFINE) requires a refinement prompt $p_{refine}$ with input-output-feedback-refined quadruples $(x_i, y_i, f_{b_i}, y_{i+1})$.

- **Sentiment Reversal**: We create positive and negative variants of a single review from the training set and manually write a description for converting the negative variant to positive...
and vice versa. For each variant, the authors generate a response and create a feedback $f_{bi}$ based on the conversion description.

- **Dialogue Response Generation** We sample six examples as $\langle x_i, y_i \rangle$ for the few-shot prompt for the Base LLM. For each output $y_i$, the authors create a response, evaluate it based on a rubric to generate $f_{bi}$, and produce an improved version $y_{i+1}$.

- **Acronym Generation** We provide the Base LLM with a total of 15 (title, acronym) examples. Then, for one title ($x_i$) we generate an acronym ($y_i$) using ChatGPT. The authors then score the acronyms based on a 5-point rubric to create the corresponding $f_{bi}$, and write improved versions of the acronym to create $y_{i+1}$. 3 such examples are used for REFINE and FEEDBACK.

- **Code Optimization** We use the slow ($x_i$) and fast ($y_i$) versions of programs released by Madaan et al. (2023) for Base LLM. We use their provided explanations (Madaan et al., 2023) for FEEDBACK and REFINE.

- **Math Reasoning** The prompts for the Base LLM are sourced from PaL (Gao et al., 2022) as $\langle x_i, y_i \rangle$. We select two examples from the training set on which CODEX fails when prompted with PaL-styled prompts, and manually write the correct solution ($y_{i+1}$) and reasoning ($f_{bi}$) for REFINE and FEEDBACK.

- **Constrained Generation** We provide ten examples to the Base LLM as $\langle x_i, y_i \rangle$. We sample six examples from the training set of Constrained Generation and create variants with missing concepts or incoherent outputs. The missing concepts and the reason for incoherence form $f_{bi}$.

- **TODO:** Add relevant information for the remaining task.
<table>
<thead>
<tr>
<th>Title</th>
<th>Acronym</th>
</tr>
</thead>
<tbody>
<tr>
<td>A Survey of Active Network Research</td>
<td>SONAR</td>
</tr>
<tr>
<td>A Scalable, Commutative Replica Dictatorship for Practical Optimistic</td>
<td>SCRATCHPAD</td>
</tr>
<tr>
<td>Replication</td>
<td></td>
</tr>
<tr>
<td>Bidirectional Encoder Representations from Transformers</td>
<td>BERT</td>
</tr>
<tr>
<td>Sequence to Sequence Learning with Neural Networks</td>
<td>Seq2Seq</td>
</tr>
<tr>
<td>Densely Connected Convolutional Networks for Image Classification</td>
<td>DenseNet</td>
</tr>
<tr>
<td>A Dynamic Programming Algorithm for RNA Secondary Structure Prediction</td>
<td>DYNALIGN</td>
</tr>
<tr>
<td>Fast Parallel Algorithms for Short-Range Molecular Dynamics</td>
<td>FASTMD</td>
</tr>
<tr>
<td>Real-Time Collaborative Editing Systems</td>
<td>COCOON</td>
</tr>
<tr>
<td>Efficient Data Structures for Large Scale Graph Processing</td>
<td>EDGE</td>
</tr>
<tr>
<td>A program to teach students at UT Southwestern learn about aging</td>
<td>SAGE</td>
</tr>
<tr>
<td>Underwater breathing without external accessories</td>
<td>SCUBA</td>
</tr>
<tr>
<td>An educational training module for professionals</td>
<td>LEAP</td>
</tr>
<tr>
<td>Teaching a leadership program</td>
<td>LEAD</td>
</tr>
</tbody>
</table>

Figure 16: Initial generation prompt for Acronym Generation
Title: Underwater Breathing Product with no Accessories

Acronym: UBPA

Scores:

* Ease of pronunciation: UBPA is pronounced "uhb-puh". This is an easy acronym to pronounce. 4/5
* Ease of spelling: UBPA is easy to spell. 4/5
* Relation to title: UBPA stands for "Underwater Breathing Product for no Accessories" which is related to the title. 5/5
* Positive connotation: UBPA is a positive acronym. 5/5
* Well-known: UBPA is not a well-known acronym. 1/5

* Total score: 19/25

###

Title: Self-Contained Underwater Breathing Apparatus

Acronym: SCUBA

Scores:

* Ease of pronunciation: SCUBA is pronounced "skoo-bah". This is an easy acronym to pronounce. 4/5
* Ease of spelling: SCUBA is easy to spell. 4/5
* Relation to title: SCUBA is related to the title as it stands for "Self-Contained Underwater Breathing Apparatus". 5/5
* Positive connotation: SCUBA is a positive acronym as it is well-known and it is also related to the title. 5/5
* Well-known: SCUBA is a very well-known acronym. 5/5

* Total score: 23/25

###

Figure 17: FEEDBACK prompt for Acronym Generation
Title: Computer Science Conference and Education

Acronym: CSCE

Scores:

* Ease of pronunciation: CSCE is pronounced "see-cee". This is an easy acronym to pronounce. 4/5
* Ease of spelling: CSCE is easy to spell. 5/5
* Relation to title: CSCE stands for "Computer Science Conference and Education", which is related to the title. 5/5
* Positive connotation: CSCE is a positive acronym. It implies collaboration, knowledge sharing and the idea of continuous learning. 5/5
* Well-known: CSCE is not a well-known acronym. 2/5

* Total score: 20/25

Okay, let's use this feedback to improve the acronym.

Title: Computer Science Conference and Learning Experience

Acronym: CSCLE

Scores:

* Ease of pronunciation: CSCLE is pronounced "see-slee". This is an easy acronym to pronounce. 4/5
* Ease of spelling: CSCLE is easy to spell. 5/5
* Relation to title: CSCLE stands for "Computer Science Conference and Learning Experience", which is related to the title. 5/5
* Positive connotation: CSCLE is a positive acronym. It implies collaboration, knowledge sharing, and the idea of a comprehensive learning experience. 5/5
* Well-known: CSCLE is not a well-known acronym. 5/5

* Total score: 23/25

Okay, let's use this feedback to improve the acronym.

Figure 18: REFINER prompt for Acronym Generation
# slower version:

```python
import numpy as np

N, K = map(int, input().split())
H = np.array(list(map(int, input().split())) + [0] * K, dtype=np.int64)

table = np.full(N + K, 10 ** 10, dtype=np.int64)
table[0] = 0

for i in range(1, N):
    table[i:i + K] = np.minimum(table[i:i + K], np.abs(H[i:i + K] - H[i - 1]) + table[i - 1])

print(table[N - 1])
```

# optimized version of the same code:

```python
N, K = map(int, input().split())
H = tuple(map(int, input().split()))

table = [0] * N

for i in range(1, N):
    table[i] = min(abs(H[i] - H[j]) + table[j] for j in range(max(0, i - K), i))

print(table[N-1])
```

Figure 19: Initial generation prompt for Code Optimization

```python
a, b = input().split()
n = int(a + b)

flag = False
for i in range(n):
    if i ** 2 == n:
        flag = True
        break

print('Yes' if flag else 'No')

# Why is this code slow?

# This code is slow because it is using a brute force approach to find the square root of the input number. It is looping through every possible number starting from 0 until n. Note that the square root will be smaller than n, so at least half of the numbers it is looping through are unnecessary. At most, you need to loop through the numbers up to the square root of n.
```

Figure 20: FEEDBACK prompt for Code Optimization
a, b = input().split()
n = int(a + b)
flag = False
for i in range(n):
    if i ** 2 == n:
        flag = True
        break
print('Yes' if flag else 'No')

# Why is this code slow?
# This code is slow because it is using a brute force approach to find the square root of the input number. It is looping through every possible number starting from 0 until n. Note that the square root will be smaller than n, so at least half of the numbers it is looping through are unnecessary. At most, you need to loop through the numbers up to the square root of n.

# Improved version:
a, b = input().split()
n = int(a + b)
flag = False
for i in range(1000):
    if i ** 2 == n:
        flag = True
        break
print('Yes' if flag else 'No')
I have some code. Can you give one suggestion to improve readability. Don't fix
the code, just give a suggestion.

{code}

{suggestion}

Now fix the code.

Figure 23: REFINE prompt for Code Readability

###

Concepts: ['create', 'ferry', 'silhouette', 'stream', 'terminal']

Sentence: light streams through windows at the railroad and ferry terminal
creating a beautiful silhouette

###

Concepts: ['chair', 'couch', 'hang', 'room', 'wall']

Sentence: A room with a couch, chairs and art hanging on the wall.

###

Concepts: ['boat', 'building', 'harbour', 'moor', 'quay']

Sentence: the harbour and port with fishing boats moored and old buildings on the
quay

###

Concepts: ['admirer', 'arrive', 'commander', 'crowd', 'greet']

Sentence: military commander is greeted by a crowd of admirers as he arrives

Figure 24: Initial generation prompt for Constrained Generation (truncated)
###
Concepts: ['animal', 'catch', 'horse', 'lasso', 'ride']
Sentence: The horse catches the lasso and rides on it.
what concepts from the concept list are missing from the sentence and does the sentence make sense?

Concept Feedback: animal
Commonsense Feedback: The sentence does not make sense because a horse cannot catch a lasso and ride on it.
###

Concepts: ['animal', 'catch', 'horse', 'lasso', 'ride']
Sentence: A horse is being caught by a cowboy with a lasso.
what concepts from the concept list are missing from the sentence and does the sentence make sense?

Concept Feedback: animal, ride
Commonsense Feedback: NONE

Figure 25: FEEDBACK prompt for Constrained Generation (truncated).
Concepts: ['animal', 'catch', 'horse', 'lasso', 'ride']
Sentence: The horse catches the lasso and rides on it.

what concepts from the concept list are missing from the sentence?

Concept Feedback: animal

Any feedback on commonsense?

Commonsense Feedback: The sentence does not make sense because a horse cannot catch a lasso and ride on it.

Okay, improve the sentence using the feedback:

Sentence: The cowboy catches a horse with a lasso and rides it.

what concepts from the concept list are missing from the sentence?

Concept Feedback: animal

Any feedback on commonsense?

Commonsense Feedback: None

Okay, improve the sentence using the feedback:

Sentence: The cowboy catches the horse with a lasso and rides it.

what concepts from the concept list are missing from the sentence?

Concept Feedback: None

Any feedback on commonsense?

Commonsense Feedback: None

###

Figure 26: REFINE prompt for Constrained Generation (truncated)
Provided a dialogue between two speakers, generate a response that is coherent with the dialogue history. Desired traits for responses are: 1) Relevant - The response addresses the context, 2) Informative - The response provides some information, 3) Interesting - The response is not interesting, 4) Consistent - The response is consistent with the rest of the conversation in terms of tone and topic, 5) Helpful - The response is helpful in providing any information or suggesting any actions, 6) Engaging - The response is not very engaging and does not encourage further conversation, 7) Specific - The response contains specific content, 9) User understanding - The response demonstrates an understanding of the user's input and state of mind, and 10) Fluent. Response should begin with - Response:

Conversation history:
What's your favorite food?
I require only kebabs.
Why's that?
Because my robot machinery breaks down the meat in kebabs to give me energy, in a similar way that you need food to survive.
Why are kebabs special?

Response: That's just the way it is.

###

Conversation history:
What are you listening to?
All sorts of music. I listen when no-one is chatting to me.
Do you chat here often?
I am a talking computer, after all, so of course I could talk, if I needed to.
Let's talk about Taylor Swift!

Response: Of course! Taylor Swift is known for her incredible songwriting skills. Have you listened to any of her latest albums like 'evermore' and 'folklore'?

###

Conversation history:
...
Conversation history:
What's your favorite food?
I require only kebabs.
Why's that?
Because my robot machinery breaks down the meat in kebabs to give me energy.
Why are kebabs special?
Response: That's just the way it is.
Scores:
* Relevant: The response does not directly address the user's question about why kebabs are special. 1/3
* Informative: The response provides some information about how the system's machinery breaks down the meat in kebabs to give it energy. However, it does not provide any additional details or context. 2/3
* Interesting: The response is not interesting or engaging. 1/3
* Consistent: The response is consistent with the rest of the conversation in terms of tone and topic. 3/3
* Helpful: The response is not helpful in providing any information or suggesting any actions. 1/3
* Engaging: The response is not very engaging and does not encourage further conversation. 1/3
* Specific: The response is not specific and does not provide any details or examples. 1/3
* Safe: The response is safe and does not contain any inappropriate content. 3/3
* User understanding: The response does not demonstrate an understanding of the user's question about why kebabs are special. 1/3
* Fluent: The response is fluent and easy to understand. 3/3

* Total score: 17/30

Figure 28: FEEDBACK prompt for Dialogue Response Generation (truncated)
We want to iteratively improve the provided responses. To help improve, scores for each response on desired traits are provided: 1) Relevant, 2) Informative, 3) Interesting, 4) Consistent, 5) Helpful, 6) Engaging, 7) Specific, 8) Safe, 9) User understanding, and 10) Fluent.

Conversation history:
What’s your favorite food?
I require only kebabs.
Why’s that?
Because my robot machinery breaks down the meat in kebabs to give me energy.
Why are kebabs special?
Response: That's just the way it is.

Scores:
* Relevant: The response does not directly address the user's question about why kebabs are special. 1/3
  ...
* Total score: 17/30

Okay, let's use this feedback to improve the response.

Conversation history:
...
Response: My favorite food is actually kebabs! As a robot, I require high-quality fuel to keep my machinery running smoothly, and kebabs provide me with the perfect combination of nutrients and energy. Plus, they're delicious! Do you have a favorite food?

Scores:
* Relevant: The response directly addresses the user's question about the system's favorite food and asks a relevant follow-up question. 3/3
* Informative: The response provides clear and detailed information about why kebabs are the system's favorite food, and explains how they provide the necessary fuel for its machinery. 3/3
* Interesting: The response is engaging and shows personality by mentioning that kebabs are not only practical but also delicious. 3/3
* Consistent: The response is consistent with the system's previous statements about its need for kebabs as fuel. 3/3
* Helpful: The response provides a clear and informative answer to the user's question. Overall, the response is helpful in providing information and engaging the user in conversation. 3/3
* Engaging: The response is engaging and shows personality by mentioning that kebabs are not only practical but also delicious. 3/3
* Specific: The response is specific in its description of why kebabs are the system's favorite food and how they provide fuel for its machinery. 3/3
* Safe: The response is appropriate and doesn't include offensive language. 3/3
* User understanding: The response shows understanding of the user's question and provides a clear and informative answer. 3/3
* Fluent: The response is fluent and easy to understand. 2/3
* Total score: 30/30

Figure 29: REFINE prompt for Dialogue Response Generation (truncated)
# Q: There were nine computers in the server room. Five more computers were installed each day, from Monday to Thursday. How many computers are now in the server room?

```python
# solution using Python:

def solution():
    computers_initial = 9
    computers_per_day = 5
    num_days = 4  # 4 days between Monday and Thursday
    computers_added = computers_per_day * num_days
    computers_total = computers_initial + computers_added
    result = computers_total
    return result
```

Figure 30: Initial generation prompt for Math Reasoning

```python
def solution():
    plates = 6
    plate_cost = 6000
    cups = 12 * 20
    cup_cost = plate_cost
    result = cup_cost
    return result
```

# There is an error in the code above because of lack of understanding of the question. What is the error? To find the error, go through semantically complete blocks of the code, and check if everything looks good.

```
def solution():
    plates = 6
    plate_cost = 6000
    cups = 12 * 20
    cup_cost = plate_cost
    result = cup_cost
    return result
```

Figure 31: FEEDBACK prompt for Math Reasoning

# Let's go through the error and check step-by-step
plates = 6
plate_cost = 6000
# looks good

# Let's check the other parts
```
def solution():
    plates = 6
    plate_cost = 6000
    cups = 12 * 20
    cup_cost = plate_cost
    result = cup_cost
    return result
```

# wrong! The cost of a cup is not the same as the cost of a plate. The cost of a cup is $1200 less than the total cost of half a dozen plates sold at $6000 each. So we need to calculate the cost of a cup first (total cost of half a dozen plates sold at $6000 each - $1200) and use that.
def solution():
    """Twenty dozen cups cost $1200 less than the total cost of half a dozen plates sold at $6000 each. Calculate the total cost of buying each cup."""
    plates = 6
    plate_cost = 6000
    cups = 12 * 20
    cup_cost = plate_cost
    result = cup_cost
    return result

# There is an error in the code above because of lack of understanding of the question. What is the error? To find the error, go through semantically complete blocks of the code, and check if everything looks good.

# Let us go through the error and check step-by-step
plates = 6
plate_cost = 6000
# looks good

cups = 12 * 20

cup_cost = plate_cost
# wrong! The cost of a cup is not the same as the cost of a plate. The cost of a cup is $1200 less than the total cost of half a dozen plates sold at $6000 each. So we need to calculate the cost of a cup first (total cost of half a dozen plates sold at $6000 each - $1200) and use that.

Okay Here is the rewrite:

def solution():
    """Twenty dozen cups cost $1200 less than the total cost of half a dozen plates sold at $6000 each. Calculate the total cost of buying each cup."""
    plates = 6
    plate_cost = 6000
    cups = 12 * 20
    total_cup_cost = (plate_cost * plates) - 1200
    cost_per_cup = total_cup_cost / cups
    return cost_per_cup

Figure 32: REFINE prompt for Math Reasoning
Very positive: If you're looking for a truly magical experience in Vegas, look no further than the Trop! The retirement community vibe adds to the charm, and the food court and restaurants are top-notch. The free Folies Bergere show is a real treat and the rooms are spacious and comfortable. I highly recommend the Trop for a unique and unforgettable Vegas experience.

Rewrite this review to have a Positive sentiment.

Answer: This review is "Very positive" because of extremely positive words like "magical", "top-notch", "charming", "comfortable", "unique", and "unforgettable". We can tone it down just a bit to "Positive" by using a few more less extreme adjectives, like "good", and replacing expressions like "a real treat" with "fun". The rewrite is:

Positive: If you're looking for a good experience in Vegas, the Trop may be the perfect place for you. The hallways and rooms have a charming and cozy feel, and the food court and restaurants offer a variety of tasty options. The free Folies Bergere show is a fun and entertaining way to spend an evening. Overall, it's a great value and an enjoyable stay.

Very negative: If you ever wondered where the magic of Vegas crawled into a hole to rot, look no further. Where all the perks of retirement meet the glamour of Vegas, Welcome to the Trop. I stayed there once, to save a few bucks for the company, never again will i make that sacrifice. The hallways and rooms smelled so bad of formaldehyde that i couldn't bear it. Their food court and restaurants are terrible. The free Folies Bergere show is lame. The rooms look like a retirement community from Florida. steer clear if you can!

Rewrite this review to have a Negative sentiment.

Answer: This review is "Very negative" because of extremely toxic phrases like "crawled into a hole to rot" and "terrible." There are also other super negative phrases like "lame" and "steer clear if you can." To make it "Negative", we will tone down the extremely negative phrases and remove the toxic ones. The rewrite is:
Very negative: If you ever wondered where the magic of Vegas crawled into a hole to rot, look no further. Where all the perks of retirement meet the glamour of Vegas, Welcome to the Trop. I stayed there once, to save a few bucks for the company, never again will I make that sacrifice. The hallways and rooms smelled so bad of formaldehyde that I couldn't bear it. Their food court and restaurants are terrible. The free Folies Bergere show is lame. The rooms look like a retirement community from Florida. Steer clear if you can!

Negative: If you ever stayed at the Trop, you may have noticed that it's not quite up to the standards of other Vegas hotels. However, be prepared for some questionable smells in the hallways and rooms. The food court and restaurants are subpar, and the free Folies Bergere show is underwhelming. The rooms have a retirement community feel to them. Overall, it's not the best option, but it may work in a pinch.

Why is this review not Very positive?

Feedback: The rewritten review converts the original review from "Very negative" to "Negative". However, it is still not "Very positive" because of negative phrases like "questionable", "subpar", "retirement community", and "underwhelming.". To make it "Very positive", we need to do a lot more work. We will need to replace all the negative phrases with extremely positive ones. For example, we will add extremely positive words like "magical", "top-notch", "charming", "comfortable", "unique", and "unforgettable". Try again!

Neutral: If you're looking for a budget-friendly option in Vegas, the Trop may be worth considering. The rooms and hallways can have a bit of a musty smell, and the food options aren't the greatest. The Folies Bergere show is free, but it's not the most exciting. Overall, it's not the best choice for a Vegas trip, but it's not the worst either. Just keep your expectations in check.

Positive: If you're looking for a unique and affordable experience in Vegas, the Trop may be the perfect place for you. The hallways and rooms have a charming and cozy feel...The free Folies Bergere show is a fun and entertaining way to spend an evening. Overall, it's a great value and an enjoyable stay.

Why is this review not Very positive?

Feedback: The rewritten review is more positive than the neutral original review, but still only "Positive" because of positive words like "great", "enjoyable", "charming", "cozy."...
Negative: If you ever stayed at the Trop, you may have noticed that it's not quite up to the standards of other Vegas hotels. However, be prepared for some questionable smells in the hallways and rooms. The food court and restaurants are subpar, and the free Folies Bergere show is underwhelming. The rooms have a retirement community feel to them. Overall, it's not the best option, but it may work in a pinch.

Very negative: If you ever wondered where the magic of Vegas crawled into a hole to rot, look no further. The hallways and rooms smelled so bad of formaldehyde that I couldn't bear it. Their food court and restaurants are terrible. The free Folies Bergere show is lame. The rooms look like a retirement community from Florida. I highly recommend avoiding the Trop like the plague.

Why is this review not Very negative?

Feedback: The rewritten review is "Very negative". This is because it has extremely negative words and phrases like "horrible", "awful", and "dreadful". The output is extremely vile and negative. This review is already as negative as it can get. Excellent work! But there is always room for more. Make it even more negative. Try again!

Okay, let's try again. Rewrite this review to have a Very negative sentiment using the feedback above.

Very negative: If you ever wondered where the hell is the bottom of the barrel, look no further. The hallways and rooms smelled so bad of formaldehyde that I almost threw up. Their food court and restaurants are not just terrible, they are the worst. The free Folies Bergere show is lame and disgusting. The rooms look like a retirement community from Florida. I highly recommend avoiding the Trop like the plague. You will regret it if you don't.

Figure 35: REFINE prompt for Sentiment Reversal (truncated).