## O Math Reasoning

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
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.

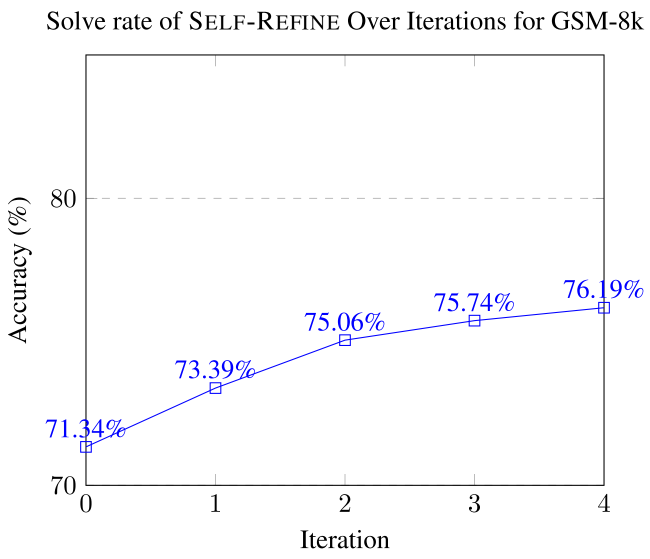

Figure 14: Improvements in accuracy on the GSM-8k math reasoning benchmark as a function of the # of iterations of SELF-REFINE.

## 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`, 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. Specifically, 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.