SELF-REFINE:
Iterative Refinement with Self-Feedback

Motivation

- In problem-solving, human performs an *iterative refinement*, where one makes an initial draft and sequentially refine it via self-feedback.
- To mimic this process with LLM, external supervision, or reward models have been utilized.
- However, such approaches require large amount of training data, or *human feedback*, which can be very expensive, or even infeasible to get.
- Can we utilize the iterative refinement with self-feedback?

Overview

(0) Generate an initial output with $\mathcal{M}$
(1) Send the output back to $\mathcal{M}$ to get a feedback
(2) Get feedback from $\mathcal{M}$
(3) Send feedback to $\mathcal{M}$ ((3))
(0) Generate a refined output with $\mathcal{M}$

$\vdots$

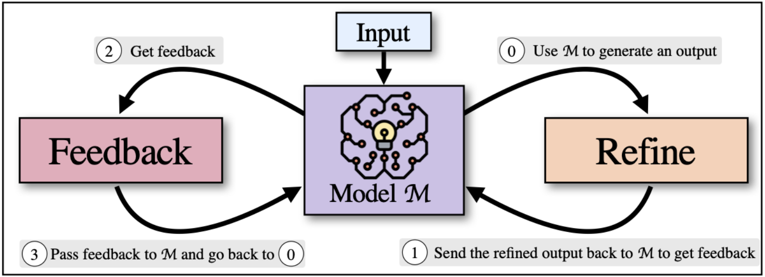

Figure 1: Overview of SELF-REFINE

Few-shot Prompting

Note in the overview, a *single* model handles initial generation, feedback, and refinement. How?
$\Longrightarrow$ *Few-shot prompting* (also called as *in-context learning*)

Trend in NLP

Past Representation learning + task-specific architecture

$\Downarrow$

Current Pretrained task-agnostic language model + direct fine-tuning

Limitation

1. Need a large dataset for every new tasks
2. Poor generalization (or spurious correlation in training data)
3. Counter-intuitive with how human mind operate

Few-shot Prompting (In-context Learning)

Train a model to develop wide range of abilities, and use those abilities to work on desired downstream task.

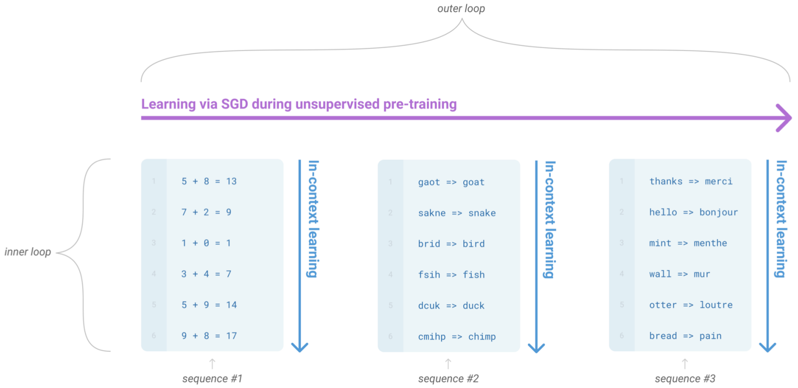

Figure 2: In-context learning

Few-shot Prompting (In-context learning)

Pretrained model receives a natural language instruction and/or a few demonstrations of the desired downstream task, and is expected to complete other instances of the task by simply predicting what comes next.

- Zero-shot : model only receives a natural language instruction.
- One-shot : model receives a natural language instruction, and a single demonstration.
- Few-shot : model receives a natural language instruction and a few (typically ≤ 10) demonstrations.

Fine-tuning

Fine-tuning

`sea otter => loutre de mer` $\leftarrow$ example 1

$\downarrow$

gradient update

$\downarrow$

`peppermint => menthe poivree` $\leftarrow$ example 2

$\downarrow$

gradient update

$\downarrow$

...

$\downarrow$

`plush girafe => girafe peluche` $\leftarrow$ example N

$\downarrow$

gradient update

$\downarrow$

`cheese =>` $\leftarrow$ prompt

Few-shot Prompting (In-context learning)

Zero-shot

`Translate Englisth to French:`	$\leftarrow$ instruction
`cheese =>`	$\leftarrow$ prompt

One-shot

`Translate Englisth to French:`	$\leftarrow$ instruction
`sea otter => loutre de mer`	$\leftarrow$ example
`cheese =>`	$\leftarrow$ prompt

Few-shot

`Translate Englisth to French:`	$\leftarrow$ instruction
`sea otter => loutre de mer`	$\leftarrow$ example 1
`peppermint => menthe poivree`	$\leftarrow$ example 2
`plush girafe => girafe peluche`	$\leftarrow$ example 3
`cheese =>`	$\leftarrow$ prompt

SELF-REFINE Framework

Feedback Receives an output, and provides feedback to improve it

Refine Refine an output based on the feedback and previously generated outputs

Iterative `Feedback` → `Refine` → `Feedback` can be applied repeatedly

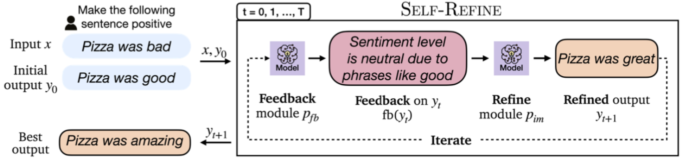

Figure 3: SELF-REFINE example

Note here the key characteristic of `Feedback` is actionable, i.e. it (i) localizes the problem, (ii) gives instruction to improve.

More Overview Examples

Sentiment Reversal

```
If you ever wondered where the magic of Vegas crawled into a
hole to rot, look no further that the Trop.  Write with positive
sentiment.

If you're looking for budget friendly option in Vegas, Trop maybe
worth considering.
```

Feedback

```
Is the sentiment of this review Positive?  If not, how can it be
improved?

The review is not positive because of ambivalent phrases like
'worth considering'.
```

Refine

```
If you're looking for a unique and affordable experience in
Vegas, the Trop may be the perfect place for you.
```

More Overview Examples

Code Optimization

Write code to generate the sum of 1, 2, $\cdots$, N

```
def sum(n):
    res = 0
                    for i in
range(n+1):
        res += 1
    return res
```

Feedback

This code is slow as it uses brute force. A better approach is to use the formula (n(n+1))/2.

Refine

```
def sum(n):
    return (n*(n+1))//2
```

SELF-REFINE Algorithm

Algorithm 1 SELF-REFINE algorithm

Given:
  input x, initial output y_0, feedback module p_{fb}, refine module p_{im}
for $t = 0, 1, \cdots, T$ **do**
  $fb, fb_{score} \sim p_{fb}(y_t)$ ▷ Get feedback
  if `stop`(fb_{score}) **then**
    break ▷ Early stopping
  else
    $y_{t+1} \sim p_{im}(y_{t+1} \mid y_{\leq t}, x, fb, fb_{score})$ ▷ Get refinement
  end if
end for
return $\arg\max_t fb_{score}(y_t)$ ▷ Best output selection

A/B Evaluation

For tasks with established metrics, evaluating the performance is easy. However, for open-ended tasks, such as Sentiment Reversal, Dialogue Response Generation, there is no reliable metrics. In this case one can use an *A/B evaluation*.

A/B Evaluation. Given an input, task instruction, and two generated outputs, a human judge blindly choose which output is better aligned with the specified instruction.

Experiments

- Math Reasoning: Solve grade school mathematics
- Constrained Generation: Given a set of concepts (or words), create a sentence that covers all the concept, and makes sense at the same time
- Code Optimization: Optimize a given program
- Code Readability: Modify a given program to improve readability
- Dialogue Response: Generate human-like response to a wide range of topics
- Sentiment Reversal: Reverse the sentiment associated with a passage
- Acronym Generation: Create an acronym

Metric	Dataset	Base LLM	SELF-REFINE
Solve Rate	Math Reasoning	71.3	**76.2**
Coverage	Constrained Generation	4.0	**22.5**
% Programs Optimized	Code Optimization	9.7	**15.6**
% Readable Variables	Code Readability	37.4	**51.3**
Human Eval.	Dialogue Response	27.2	**37.6**
	Sentiment Reversal	15.3	**84.7**
	Acronym Generation	11.8	**23.5**

Table 1: Main results

Ablation Study

1. Impact of iterative refinement

Dataset	Starting point	Iteration 1	Iteration 2
Sentiment Reversal	32.4	41.6	**84.7**
Math Reasoning	71.3	73.2	**76.2**
Code Optimization	9.7	15.3	**15.6**

2. Impact of actionable feedback

Task	Actionable	Generic
Sentiment Reversal	85%	73%
Code Optimization	15.6%	10.4%

Thank You

Q & A