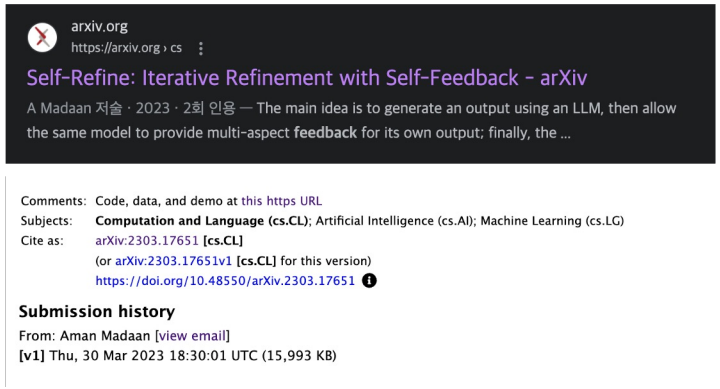
Self-Refine: Iterative Refinement with Self-Feedback

1. Language Technologies Institute, Carnegie Mellon University
2. Allen Institute for Artificial Intelligence
3. University of Washington
4. NVIDIA
5. UC San Diego
6. Google Research, Brain Team



목차

Abstract

- Large Language Model (LLM) 이 first try 에 best text 를 생성하는 것은 아님 (e.g. summaries, answers, explanations)
- **Self-Refine**
  - LLM 도 사람처럼, **생성한 text 를 refine 하는 프레임워크 제안**
  - For similarly improving initial outputs from LLMs through iterative feedback and refinement
  - LLM 을 사용하여 output 을 생성하고, 그 output 에 대해 동일 모델이 multi-aspect feedback 을 제공하여, 최종적으로 동일한 모델이 이전에 생성한 output 을 refine 한 output 을 생성할 수 있게 함
    - The main idea is to **generate an output using an LLM**, then **allow the same model to provide multi-aspect feedback** for its own output; finally, **the same model refines its previously generated output given its own feedback**
  - Supervised training data, reinforcement learning 필요 없음 (single LLM 만 사용)
- Self-Refine 을 사용했을 때
  - GPT-3.5 and GPT-4 에 대해 human, automated metrics 모두 성능 향상

Abstract

- 논문 페이지 : https://selfrefine.info/
- Demo 링크 : https://self-refine-webgen.herokuapp.com/
- 코드 링크 : https://github.com/madaan/self-refine

Introduction

- 현대 LLM 은 초기 단계에서도 output 을 잘 생성하지만, multifaceted objectives 나, less defined goals 등의 테스크에서는 여전히 불안정하기 때문에, iterative refinement 과정이 필수적임
  - To ensure that all aspects of the task are met and desired quality is achieved
  - 이를 위해 external supervision 과 significantly large training data, expensive human annotations 가 필요한 상황

- 위의 한계점을 극

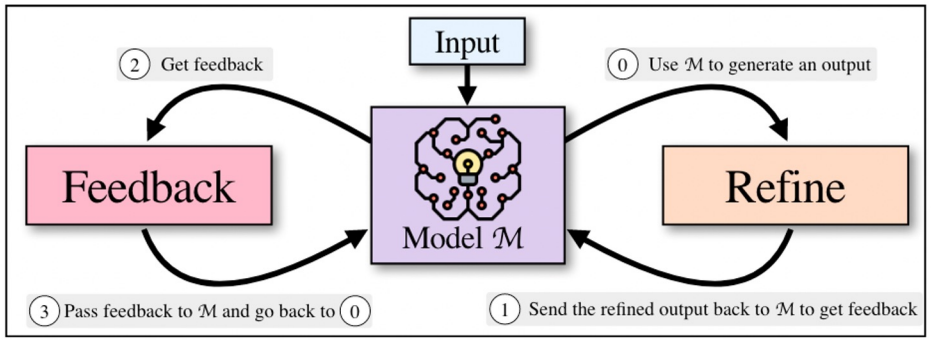


Figure 1: SELF-REFINE starts by taking an initially generated output (⓪), and passing it back to the same model $\mathcal{M}$ (①) to get feedback (②); feedback on the initial output is passed back to the model (③), to iteratively refine (⓪) the previously generated output. SELF-REFINE is instantiated with a powerful language model such as GPT-3.5 and does not involve human assistance.

Introduction

- Self-Refine 은 두 개 components 의 iterative loop 로 구성되어 있음
  - Feedback, Refine
  - 0) Model 로부터 initial draft output 생성
  - 1) Initial draft output 을 **동일한 model** 에 pass 하여 feedback 얻음
  - 2) 1)에서 얻은 feedback 을 다시 **동일한 model**에 pass 하여 refine the previously generated output
  - 위 과정을 특정 iter 만큼 반복하거나, model itself 가 더 이상의 refinement 가 불필요하다고 느껴질때까지

- The main idea in this work is that ***the same underlying language model*** ***performs both feedback and refinement*** in a few-shot setup
  - 하나의 언어 모델이 생성된 출력물에 대한 피드백을 제공하고 품질을 개선하는 데 모두 사용된다

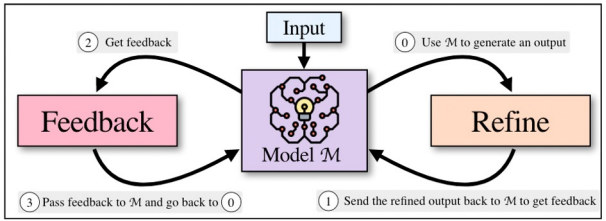


Figure 1: SELF-REFINE starts by taking an initially generated output (⓪), and passing it back to the same model $\mathcal{M}$ (①) to get feedback (②); feedback on the initial output is passed back to the model (③), to iteratively refine (⓪) the previously generated output. SELF-REFINE is instantiated with a powerful language model such as GPT-3.5 and does not involve human assistance.

Introduction

- Self-Refine 을 7개 task 에 적용
  - review rewriting, acronym generation, constrained generation, story generation, code rewriting, response generation, and toxicity removal
- A few-shot prompting
  - 모델을 따로 fine-tuning 하지 않는 대신, few-shot prompting 사용

- Contributions
  - Propose self-refine, **a novel approach that allows LLMs** to **iteratively refine their own outputs** using their own feedback, along multiple dimensions to improve performance on diverse tasks.
  - Conduct extensive experiments on **7 diverse tasks** of review rewriting, acronym generation, story generation, code rewriting, response generation, constrained generation, and toxicity removal, demonstrating that self-refine outperforms direct generation from strong generators like GPT-3.5 and even GPT-4 by at least 5%, and up to more than 40% improvement.

Related Work

- Human-generated, Machine-generated Language (NL) feedback 은 다양한 task 에 효과적임
- Feedback can be differ in the
    a. source of feedback
    b. the representation format (피드백 형태)
    c. the utilization of the feedback in generating the refined output

- **Source of feedback**
    - Human
    - Reinforcement Learning (RL)
    - Human cost 를 줄이기 위해 automated source (compilers, existing online sources like Wikipedia edits)
    - 최근에는, LLMs have been used to generate feedback for a general domain solution.
        - 그러나, 본 논문에서는 LLM 으로 generate 를 생성하는 것 뿐만 아니라, LLM 이 생성한 피드백을 반영하여, 결과물을 개선하고, 이 피드백-개선 과정을 반복하는 것을 제안함
        - 또한, 지금까지 machine-generated feedback 이 beneficial 함을 증명한 연구는 없었는데, 본 연구에서 zero-,few-shots 으로 생성된 피드백이 도움됨을 증명함
        - 추가적으로, multiple aspects 에서 피드백을 제공함

Related Work

- **Representation of the feedback**
  - **Natural Language (NL) -> 본 논문에서는 NL feedback 형태를 사용함**
    - Prescriptive
    - e.g. “replace course id with program id”
  - Non-NL
    - e.g. human-provided example pairs
    - dense signals

- **Utilization of feedback**
  - 생성된 피드백은 Supervised, RL-based refiner 를 학습시키는데 사용됨
    - Supervised from humans is costly
    - RL-based : out-of-domain generalization issue
  - 본 연구에서는 separate refiner 학습을 하지 않고, multiple domains 를 위해 few-shot LLM refiner 를 사용함

- Iterative refinement
  - 기존의 iterative (multiple times) 로 사용되던 corrector 는 어느 시점에 적절히 모델을 멈춰야 하는지 알 수 없음
  - 본 연구에서는 self-refine 이 scalar value-based stopping criteria 를 가지고 있어 위 문제를 해결함

Related Work

	Few-shot refiner	Few-shot feedback	Multi-aspect scored feedb.	Iterative framework
Learned Refiners: PEER (Schick et al., 2022b), Self-critique (Saunders et al., 2022b), Self-correct (Welleck et al., 2022),	❌	✅ or ❌	❌	✅ or ❌
RL-based: QUARK (Lu et al., 2022), RLHF (Bai et al., 2022a), CodeRL (Le et al., 2022b), Constitutional-AI (Bai et al., 2022b)	❌	❌	❌	❌
LLM-based: Self-ask (Press et al., 2022a), Augmenter (Peng et al., 2023), Re^3 (Yang et al., 2022), Reflexion (Shinn et al., 2023)	✅	✅ or ❌	❌	❌
SELF-REFINE (this paper)	✅	✅	✅	✅

Table 1: A comparison of SELF-REFINE to closely related prior approaches

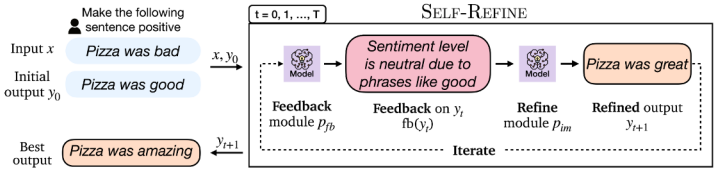
Related Work

Method	Primary Novelty	zero/few shot improvement	multi aspect critics	NL feedback with error localization	iterative framework
RLHF (Stiennon et al., 2020b)	optimize for human preference	❌trained on feedback	❌ single (human)	✅(not self gen.)	❌
Rainier RL (Liu et al., 2022)	RL to generate knowledge	❌trained on end task	❌ single(accuracy)	❌(knowl. only)	❌
QUARK RL (Lu et al., 2022)	quantization to edit generations	❌ trained on end task	❌ single(scalar score)	❌ (dense signal)	✅ (train time iter.)
Code RL (Le et al., 2022a)	actor critic RL for code improvement	❌ trained on end task	❌ single(unit tests)	❌(dense signal)	❌
DrRepair (Yasunaga and Liang, 2020)	Compiler feedback to iteratively repair	❌ trained semi sup.	❌ single(compiler msg)	✅(not self gen.)	✅
PEER (Schick et al., 2022b)	doc. edit trained on wiki edits	❌trained on edits	❌ single(accuracy)	✅(not self gen.)	✅
Self critique (Saunders et al., 2022a)	few shot critique generation	❌feedback training	❌ single(human)	✅(self gen.)	❌
Self-correct (Welleck et al., 2022)	novel training of a corrector	❌ trained on end task	❌ single (task specific)	✅(limited setting)	✅(limited setting)
Const. AI (Bai et al., 2022b)	train RL4F on automat (critique, revision) pair	❌ critique training	✅ (fixed set)	✅	❌
Self-ask (Press et al., 2022b)	ask followup ques when interim ans correct;final wrong	✅ few shot	❌ none	❌(none)	❌
GPT3 score (Fu et al., 2023)	GPT can score generations with instruction	✅few shot	❌ single(single utility fn)	❌(none)	❌
Augmenter (Peng et al., 2023)	factuality feedback from external KBs	✅few shot	❌ single(factuality)	✅(self gen.)	✅
Re^3 (Yang et al., 2022)	∼ours: but one domain, trained critics	✅few shot	✅(trained critics)	✅(not self gen.)	✅
SELF-REFINE	fewshot iterative multi aspect NL fb	✅few shot	✅ multiple(few shot critics)	✅(self gen.)	✅

Table 7: Summary of related approaches. Reinforcement learning approaches are shown in purple , trained corrector approaches are shown in orange , and few-shot corrector approaches are shown in green .

Self-Refine Framework

- Initial task 와 prompt 가 주어졌을 때, self-refine 은 first output attempt 생성
- 그리고 own output 에 대해 feedback 제공 (**Feedback**)
- Feedback 을 반영한 enhanced output 생성 (**Refine**)
- Can iteratively apply this feedback-and-refine approach for each subsequent generation, allowing for iterative refinement



Algorithm 1 SELF-REFINE algorithm

Require: input x, initial output y_0, feedback module p_{fb}, refine module p_{im}

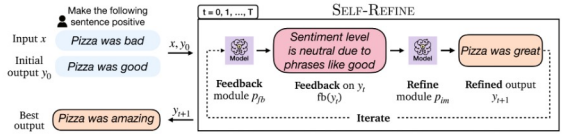
for iteration t $\in 0...T$ **do**
  $fb, fb_{score} \sim p_{fb}(y_t)$ ▷ Get Feedback
  if $\texttt{stop}(fb_{score})$ **then**
    break ▷ Early stopping
  else
    $y_{t+1} \sim p_{rf}(y_{t+1} \mid y_{\leq t}, x, fb, fb_{score})$ ▷ Get Refined output
  end if
end for
return $\arg\max_t \ fb_{score}(y_t)$ ▷ Best output selection

Figure 2: SELF-REFINE algorithm and an illustrated example. (Top) SELF-REFINE applied to Sentiment Reversal. (Bottom) our algorithm, see §3.1 for details.

Self-Refine Framework

Feedback

- Receives the initial output y_{0} and provides feedback on **how to enhance it**
- This feedback is **task-dependent** and generally addresses **multiple aspects of the input**
  - e.g. 피자 예시 - 피드백은 sentiment level 과 **vividness** (명료성) 을 중시함. 또한 제공된 피드백은 **actionable** 함. 즉, 모델이 피드백을 받고, refined 할 수 있는 수준 (e.g. the sentiment level is neutral, the sentiment is neutral due to phrases like good)



Algorithm 1 SELF-REFINE algorithm

Require: input x, initial output y_0, feedback module p_{fb}, refine module p_{im}

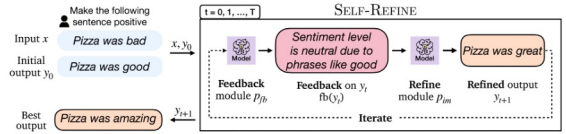
for iteration t $\in 0...T$ **do**
  $fb, fb_{score} \sim p_{fb}(y_t)$ ▷ Get Feedback
  if stop(fb_{score}) **then**
    break ▷ Early stopping
  else
    $y_{t+1} \sim p_{rf}(y_{t+1} \mid y_{\leq t}, x, fb, fb_{score})$ ▷ Get Refined output
  end if
end for
return $\arg\max_t \ fb_{score}(y_t)$ ▷ Best output selection

Figure 2: SELF-REFINE algorithm and an illustrated example. (Top) SELF-REFINE applied to Sentiment Reversal. (Bottom) our algorithm, see §3.1 for details.

Self-Refine Framework

Reine

- Responsible for refining an output y_{t} based on the **received feedback** and the **previously generated output**
  - e.g. 피자 예시 - 피드백을 받고, ‘good’ → ‘amazing’ 으로 변경



Algorithm 1 SELF-REFINE algorithm

Require: input x, initial output y_0, feedback module p_{fb}, refine module p_{im}
for iteration t $\in 0...T$ **do**
  $fb, fb_{score} \sim p_{fb}(y_t)$ ▷ Get Feedback
  if $\texttt{stop}(fb_{score})$ **then**
    break ▷ Early stopping
  else
    $y_{t+1} \sim p_{rf}(y_{t+1} \mid y_{\leq t}, x, fb, fb_{score})$ ▷ Get Refined output
  end if
end for
return $\arg\max_t \ fb_{score}(y_t)$ ▷ Best output selection

Figure 2: SELF-REFINE algorithm and an illustrated example. (Top) SELF-REFINE applied to Sentiment Reversal. (Bottom) our algorithm, see §3.1 for details.

Self-Refine Framework

Iterative improvement

- The feedback → refine → feedback loop can be applied multiple times
- iter 종료 조건 (둘 중에 하나만 만족하면 종료)
  - The number of iters can be set to a fixed number
  - Function of the feedback, a numerical fb_{score} is above a threshold
- **Key aspect of self-refine is the retention of a history of past experiences (과거 히스토리를 기억함)**
  - **즉, previous output 을 prompt 에 추가함으로써, 과거의 실수를 반복하지 않도록 함**

Actionable and multi-aspect feedback

- Actionable feedback 을 제공한다는 것이 self-refine 의 큰 기여점
- Feedback 은 두가지 측면을 고려함
  - **localization of the problem**
    - Localization may be possible by highlighting specific tokens
    - Localization 은 Sentiment Reversal task 에서 강조될 수 있는데, Acronym Generation 과 같은 테스크에서는 명시적이지 못함
  - **instruction to improve**

Self-Refine Framework

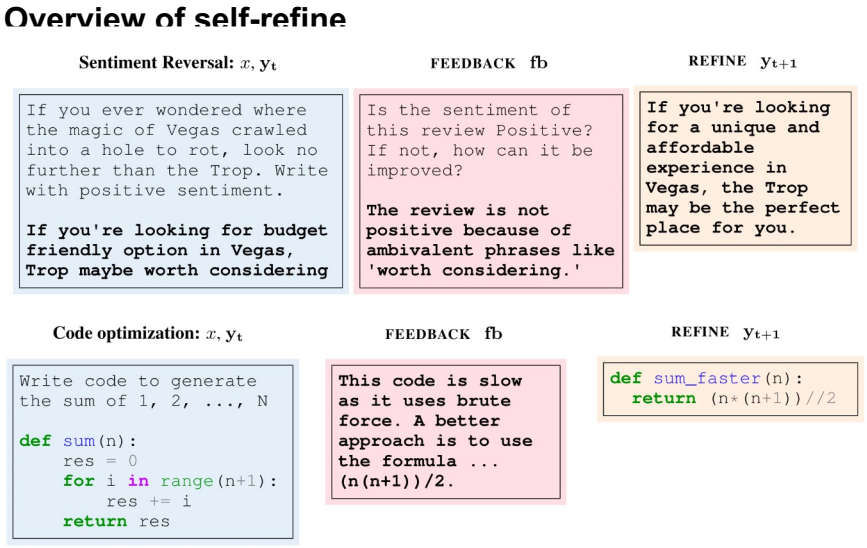


Figure 3: Overview of SELF-REFINE: given an initial output (left,), FEEDBACK evaluates it and generates actionable feedback required to correct it (center,). The REFINE takes the feedback into account, and refines the output (right,). For example, in the top row, an initial review with negative sentiment is first transformed into a positive one, then further refined through feedback. In the bottom row, an initial code snippet is provided, followed by feedback identifying a more efficient approach, and finally resulting in an optimized code implementation after applying the suggested improvements.

Instantiating Self-refine

- Both feedback and refine are implemented as **few-shot prompts** and a pretrained large language model (LLM)

Experimental Setup

Task

- 7개의 다양한 테스크 수행
  - Acronym generation , hard version of the CommonGen (Constrained Generation) 은 새롭게 소개하는 2개 테스크, 나머지 5개는 기존에 있는 테스크
- 각 테스크 별 prompt 공개
  - Initial - Feedback - Refine

Task and Description	Sample one iteration of FEEDBACK-REFINE
Sentiment Reversal Rewrite reviews to reverse sentiment. Dataset: (Zhang et al., 2015) 1000 review passages	x: The food was fantastic...” y_t: The food was disappointing...” fb: Increase negative sentiment y_{t+1}: The food was utterly terrible...”
Dialogue Response Generation Produce high-quality conversational responses. Dataset: (Mehri and Eskenazi, 2020) 372 convers.	x: What’s the best way to cook pasta?” y_t: The best way to cook pasta is to...” fb: Make response relevant, engaging, safe y_{t+1}: Boil water, add salt, and cook pasta...”
Acronym Generation Generate acronyms for a given title Dataset: (§Appendix G) 250 acronyms	x : Radio Detecting and Ranging” y_t: RDR fb : be context relevant; easy pronunciation y_{t+1}: RADAR”
Code Optimization Enhance Python code efficiency Dataset: (Madaan et al., 2023): 1000 programs	x: Nested loop for matrix product y_t: NumPy dot product function fb: Improve time complexity y_{t+1}: Use NumPy’s optimized matmul function
Code Readability Improvement Refactor Python code for readability. Dataset: (Puri et al., 2021) 300 programs*	x: Unclear variable names, no comments y_t: Descriptive names, comments fb: Enhance variable naming; add comments y_{t+1}: Clear variable names, meaningful comments
Math Reasoning Solve math reasoning problems. Dataset: (Cobbe et al., 2021) 1319 questions	x: Olivia has $23, buys 5 bagels at $3 each” y_t: Solution in Python fb: Show step-by-step solution y_{t+1}: Solution with detailed explanation
Constrained Generation Generate sentences with given keywords. Dataset: (Lin et al., 2020) 200 samples	x: beach, vacation, relaxation y_t: During our beach vacation... fb: Include keywords; maintain coherence y_{t+1}: Our beach vacation was filled with relaxation

Table 2: The diverse set of tasks to evaluate SELF-REFINE, along with their associated datasets and sizes. On the right, we show an example of a single iteration of refinement of input from the dataset x, previous generated output y_t, feedback generated fb, and refinement produced y_{t+1} using fb. Few-shot prompts used for FEEDBACK and REFINE are in §Appendix I. *: We randomly select 300 functionally-correct Python snippets.

Experimental Setup

- 각 테스크 별 prompt 공개 (Code Optimization)

```
# slower version:

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:

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 13: Initial generation 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 sqare 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 14: 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 sqare 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')
```

Figure 15: REFINE prompt for Code Optimization

Experimental Setup

Models

- Textural generation tasks
  - text-davinci-003
  - GPT-3.5
- Code-related tasks
  - code-davinci-002
- Acronym generation
  - GPT-4

- Baseline : Self-refine 이 사용되지 않은 LLM

Experimental Setup

Evaluation Metrics

- 테스크를 두 타입으로 분류
  - **Open-ended tasks (without reliable automated metrics)**
    i. Sentiment Reversal, Dialogue Response Generation, Acronym Generation, Code Readability Improvement
    ii. A/B evaluation conducted by human judges
  - **Established and representative metrics**
    i. Math reasoning, Constrained Generation, Code Optimization
    ii. Automated metrics

Results

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 3: **Main results:** relative improvements in performance across diverse tasks using the SELF-REFINE framework. The improvements are measured as the percentage increase in preference rate revealed by human evaluation ("Human Eval.") or task-specific performance metrics. This demonstrates the effectiveness of the iterative refinement process employed by SELF-REFINE in generating higher-quality outputs.

Analysis

- 논문의 method는 두 가지 primary components 에 기반함
  - iterative generation
  - feedback

- Conduct ablation studies to evaluate the following:
  - Quality of the feedback module : Overall performance 의 impact 분석
  - Quality of the refine module : 새로운 문장 생성 퀄리티 향상 조사
  - Impact of iterations : iter count 가 성능에 영향을 미치는 정도 조사

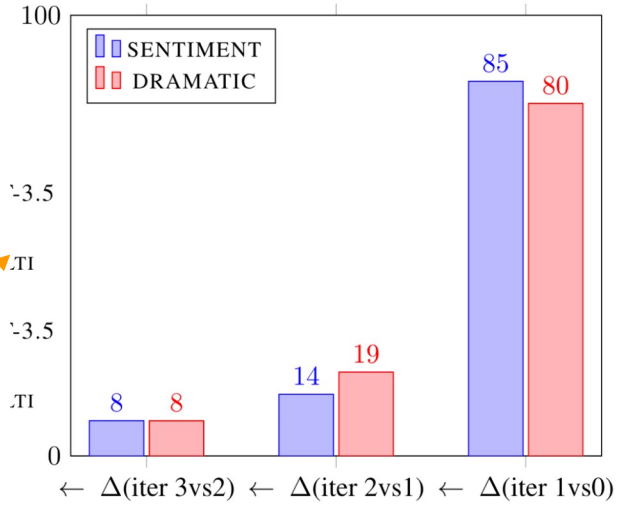
Analysis

Impact of Iterative Refinement

- Does the output improve at each step?
- Dataset (3개 테스트) : Sentiment Reversal, Math Reasoning, Code Optimization

Dataset	Starting point y_0	Iteration 1	Iteration 2	Rate of Refinement
Sentiment Reversal	32.4	41.6	**84.7**	26.15
Math Reasoning	71.3	73.2	**76.2**	2.45
Code Optimization	9.7	15.3	**15.6**	2.95

Table 4: SELF-REFINE iteratively improves outputs. The rate of Refinement represents the average improvement in percentage points per iteration, calculated from Iteration 0 to Iteration 2.



*Preference for the outputs generated by SELF-REFINE in terms of alignment with the **target sentiment** and **dramatic nature of responses** 차이가 뭔지 모르겠음*

Analysis

Non-monotonic increase in output quality for acronym generation

- acronym generation output 퀄리티가 일정한 패턴 없이 왔다갔다 한다.
- Acronym generation 은 multi-aspect feedback 을 제공하기 때문에, the output quality can fluctuate during the iterative process, improving on one aspect while losing out on another

Iteration	Acronym	Pronunciation	Pron. (5)	Spell. (5)	Rel. (5)	Pos. Con. (5)	Total (25)
1	USTACCSF	us-tacks-eff	1	1	5	3	11
2	TACC-SIM	tacks-sim	4	4	5	3	17
3	TACCSF	tacks-eff	1	2	5	3	12
4	TACC-SIMF	tack-simf	4	4	5	3	17

Table 5: 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.

Analysis

어떤 방식으로 actionable feedback 을 제공하는지는 안 나옴

Role of Actionable Feedback

- Providing detailed and informative feedback that guides the iterative refinement process
- Results of actionable vs. generic feedback ("something is wrong")

Task	Actionable Feedback	Generic Feedback	Improvement
Sentiment Reversal	85%	73%	12%
Code Optimization	15.6%	10.4%	5.2%

Table 6: Results of the ablation study, showing the percentage of preference rate for outputs generated by SELF-REFINE with informative feedback and generic feedback, as well as the percentage improvement achieved with actionable feedback over generic feedback.

Analysis

Best-of-k evaluation

- Compare SELF-REFINE against the base generator model in generating improved outputs
- Baseline model 와 self-refine 모델을 1:1로 비교한게 아니라, k:1 로 비교하였음에도 불구하고, self-refine 모델이 더 좋은 평가를 받음

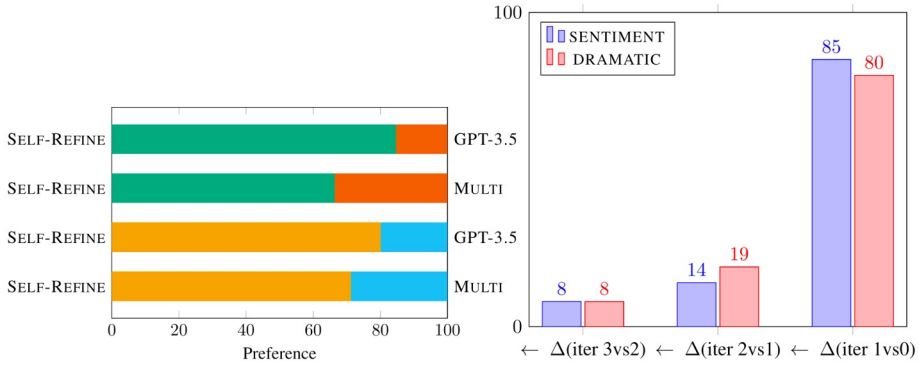


Figure 4: Preference for the outputs generated by SELF-REFINE in terms of alignment with the target sentiment (SELF-REFINE vs. baselines) and dramatic nature of responses (SELF-REFINE vs. baselines). The preference rate measures the instances where the output from a given method was judged to be more aligned with the desired sentiment level compared to SELF-REFINE. Output generated by SELF-REFINE were overwhelmingly preferred against the outputs of GPT-3.5, and the challenging setup where $k = 4$ independent samples were taken and **all of them** were compared against SELF-REFINE (1 vs. k eval). The figure on the right shows the improvement in output quality between subsequent iterations. Outputs improve in quality as iterations proceed, but the most gains come in the initial iterations.

Conclusion

- Present a novel framework that enables LLM to utilize iterative refinement and self-assessment for generating higher-quality outputs
- Self-refine operates within a single LLM, requiring neither additional training data nor reinforcement learning

Thank You