Niket Tandon, Aman Madaan, Peter Clark, and Yiming Yang. 2022. Learning to repair: Repairing model output errors after deployment using a dynamic memory of feedback. In *Findings of the Association for Computational Linguistics: NAACL 2022*, pages 339–352.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. 2022. Chain of Thought Prompting Elicits Reasoning in Large Language Models. *arXiv preprint arXiv:2201.11903*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2022. Generating sequences by learning to self-correct. *arXiv preprint arXiv:2211.00053*.

Kevin Yang, Nanyun Peng, Yuandong Tian, and Dan Klein. 2022. Re3: Generating longer stories with recursive reprompting and revision. In *Conference on Empirical Methods in Natural Language Processing*.

Michihiro Yasunaga and Percy Liang. 2020. Graph-based, self-supervised program repair from diagnostic feedback. *37th Int. Conf. Mach. Learn. ICML 2020*, PartF168147-14:10730–10739.

Xiang Zhang, Junbo Zhao, and Yann LeCun. 2015. Character-level convolutional networks for text classification. *Advances in neural information processing systems*, 28.

A Evaluation Tasks

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

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 rich conversational responses. Dataset: (Mehri and Eskenazi, 2020) 372 conv.	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..."
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 variables, 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
Acronym Generation Generate acronyms for a given title Dataset: (Appendix Q) 250 acronyms	x : Radio Detecting and Ranging" y_t: RDR fb : be context relevant; easy pronunciation y_{t+1}: RADAR"
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}: .. beach vacation was filled with relaxation

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_t, the feedback generated fb_t, and the refinement y_{t+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. (2022a)). 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.

Method	Primary Novelty	zero/few shot improvement	multi aspect critics	NL feedback with error localization	iterative framework
RLHF (Stiennon et al., 2020)	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., 2022)	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 5: 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.