

IMPROVING LANGUAGE MODEL SELF-CORRECTION CAPABILITY WITH META-FEEDBACK

Anonymous authors

Paper under double-blind review

ABSTRACT

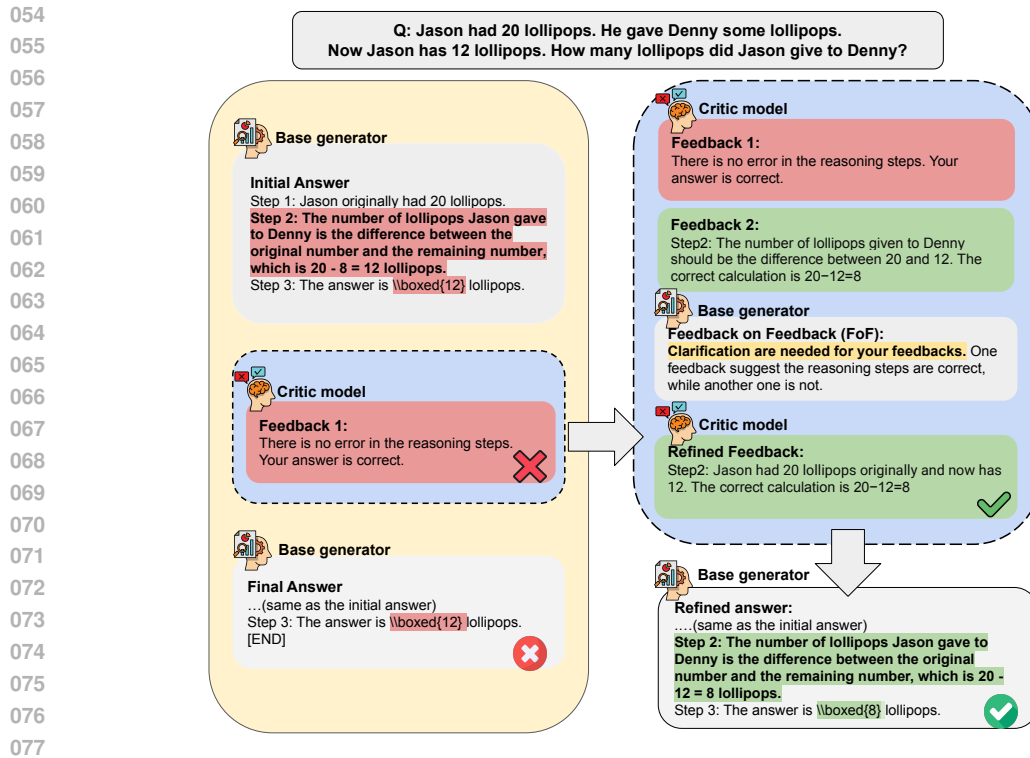
Large language models (LLMs) are capable of self-correcting their responses by generating feedback and refining the initial output. However, their performance may sometimes decline following self-correction, either because the feedback contains errors or due to unnecessarily attempting to refine an already accurate response. To address these limitations, we investigate whether the same LLM can generate *meta*-feedback that pinpoints errors in the feedback rather than the response, an ability that remains under-explored despite extensive research on LLMs’ self-feedback generation. We design a novel self-correction prompting framework, **Feedback-on-Feedback (FoF)**, which leverages meta-feedback to improve the feedback before refining the response. Our framework first samples multiple pieces of feedback for the initial response, and prompts the LLM to generate a meta-feedback that analyzes the inconsistency between these feedback. Based on the meta-feedback, the LLM generates refined feedback that subsequently guides the revision of the response. Our FoF framework consistently outperforms competitive baselines across two LLMs on three datasets, covering arithmetic reasoning, machine translation, and programming tasks. Specifically, FoF improves performance on GSM8K by 3.6 points (45.2% vs. 41.6% for the initial answer) and on MBPP by 6.4 points (51.7% vs. 45.3%) using the LLaMA-3-8B model.

1 INTRODUCTION

LLMs have revolutionized the field of natural language processing, demonstrating exceptional performance across various tasks such as language generation, translation, and question answering (OpenAI et al., 2024). Despite their remarkable capabilities, LLMs often struggle with producing consistently accurate, coherent, and contextually relevant responses (Madaan et al., 2023; Chen et al., 2023b; Welleck et al., 2022). A critical area for improvement in LLMs is their intrinsic ability for self-correction—the capacity to identify and fix errors, inconsistencies, or shortcomings in their outputs without relying on external feedback, programs, or knowledge bases (Pan et al., 2023; Madaan et al., 2023; Chen et al., 2023b). This process typically involves the model first generating a critique that identifies the limitations of its initial response, followed by revising the response based on the self-generated critique. The critique-revise process can be iterated multiple times to progressively refine the model’s output, allowing for a more thorough and comprehensive self-correction (Madaan et al., 2023; Shinn et al., 2023; Kim et al., 2023).

Many existing methods typically rely on external feedback or oracle labels (Madaan et al., 2023; Huang et al., 2024), which are often unavailable during inference. To address this, another line of research dives into the *intrinsic* self-correction ability (Huang et al., 2024) of LLMs to refine the answer without access to external information and oracle labels. However, they suggest that intrinsic self-correction harms the model performance in reasoning tasks since LM struggles to determine the correctness of the initial answer, leading to revising an answer that is already correct. Besides, the quality of LLM-generated feedback can be arbitrarily bad without proper guidance or selection, leading to inferior performance (Shridhar et al., 2023; Liang et al., 2023b). This motivates us to improve the quality of the feedback before applying it to refine the answer.

Particularly, LLMs’ capacities to provide constructive feedback on their own self-feedback, known as meta-feedback (Lan et al., 2024), remains less explored. Lan et al. (2024) investigate this concept by prompting LLMs to critique the quality of their own feedback. While the study demonstrates



078
079
080
081
082
083
084
085
086
087
088
089
090

Figure 1: An illustrative example of FoF compared to the Self-refine setting (Madaan et al., 2023). The question is from GSM8K (Cobbe et al., 2021a), and all answers and feedback are generated by GPT-3.5-turbo-0515. In the Self-refine setting (left), the base generator produces an initial answer, and the critic model provides feedback on it. However, since the feedback is incorrect, the answer remains a wrong answer. In the FoF setting (right), two (or multiple) pieces of feedback responses are sampled from the critic model. The base generator recognizes the conflict between the feedback, prompting the critic model to clarify and correct it. Based on the refined feedback, the answer model updates the wrong answer to provide the correct answer.

091
092
093
094
095
096

that models like GPT-4 can generate meta-feedback, the findings also highlight significant limitations—LLMs struggle with consistency and accuracy of feedback, especially in complex tasks such as mathematical reasoning and coding. The quality of meta-feedback often lags behind human, indicating room for improvement in LLMs’ ability to self-evaluate. In this paper, we study the research question: **Can the meta-feedback improve the quality of feedback generated by LLMs, and subsequently enhance the final output?**

097
098
099
100
101
102
103
104
105
106
107

To this end, we propose a Feedback-on-Feedback (FoF) framework. Inspired by self-consistency approaches, we focus on the consistency of self-generated feedback by LLMs. Specifically, we explore how identifying and resolving inconsistencies in self-generated feedback can improve the quality of the final output. Unlike methods that rely on external feedback or oracle labels, the FoF framework samples multiple feedback and then identifies inconsistencies between multiple LLM self-generated feedback based on their semantic similarities. Then, FoF 1) generates additional meta-feedback to analyze these inconsistencies, 2) refines the feedback with the meta-feedback, and 3) revises the answers using the refined feedback. An example of how FoF works is demonstrated in Figure 1. When the first feedback indicates the initial answer is correct and the second feedback shows there is still an error in the answer, combining different stances of feedback and the clarification from meta-feedback together provides more accurate feedback. This approach enables FoF to operate effectively in zero-shot scenarios without demonstrations, highlighting its generalizability across various tasks.

We conduct experiments on three datasets: GSM8K (arithmetic reasoning) (Cobbe et al., 2021a), CSMT (machine translation) (He et al., 2020), and MBPP (programming problem-solving) (Austin et al., 2021). Our FoF method outperforms the Self-Refine (Madaan et al., 2023), the Self-Consistency (Wang et al., 2023c) baseline and zero-shot CoT prompt (Wei et al., 2023) across all tasks and two models including one close-source model—GPT-3.5-0515 (Brown et al., 2020) and an open-source model—LLaMA3-8B (Touvron et al., 2023) in the zero-shot setting. Notably, FoF achieves an average improvement of 3.54 points across GSM8K, CSMT, and MBPP tasks compared to Self-Refine using LLaMA-3-8B. These results demonstrate the effectiveness of the FoF method in enhancing the self-correction ability of LLMs across various tasks and model sizes. Additionally, our ablation studies show that the quality of the critic model is key to the FoF framework’s effectiveness. Using a more advanced critic like GPT-4 improves critiques and guides the base generator better.

Our contributions are threefold:

1. We introduce the **Feedback-on-Feedback (FoF)** framework to enhance language model self-correction capability by aggregating over multiple pieces of self-feedback to generate more accurate feedback that guides answer revision.
2. We conduct experiments on GSM8K, CSMT, and MBPP using GPT-3.5-0515 and LLaMA-3-8B. The results demonstrate improvements of up to 6.4 percentage points on MBPP compared to the initial answer using the LLaMA-3-8B model.
3. We highlight the importance of selecting and integrating multiple feedback to improve answer accuracy. Our approach ensures more accurate and consistent self-correction by addressing inconsistencies between feedback.

2 RELATED WORKS

Natural Language Feedback The ability of LLMs to self-correct has garnered significant attention, with various approaches proposed to enhance this capability. Recent advancements leverage model natural language feedback and iterative refinement techniques (Ye et al., 2023; Madaan et al., 2023; Shinn et al., 2023; Kim et al., 2023). Approaches include iterative refinement through feedback alignment (Madaan et al., 2023; Gou et al., 2024; Ye et al., 2023), reinforcement learning for feedback optimization (Akyurek et al., 2023; Shinn et al., 2023; Kumar et al., 2024), and using external evaluation metrics to guide self-correction (Aggarwal et al., 2023; Paul et al., 2024; Zheng et al., 2023; Kim et al., 2023). Other methods integrate diverse prompts and verifiers, such as using self-verifiers or external verifiers to score reasoning paths (Gero et al., 2023; Li et al., 2023c; Zelikman et al., 2022; Cobbe et al., 2021b; Weng et al., 2023; Zhang et al., 2024), and multi-agent debate systems where LLMs interact to reach a consensus (Du et al., 2023; Cohen et al., 2023; Li et al., 2023a; Liang et al., 2023a). Notably, Kamoi et al. (2024b) highlights that LLMs can effectively self-correct under conditions such as task suitability, reliable feedback sources, model fine-tuning, strong self-evaluation mechanisms, and iterative feedback loops during inference, while Valmeekam et al. (2023) question LLMs’ ability to self-critique effectively in planning tasks, further demonstrating the limitations of such frameworks.

However, some of the methods (Shinn et al., 2023; Madaan et al., 2023; Kim et al., 2023) depend on oracle labels or external feedback to determine when to stop the self-correction process. Multi-agent debate settings have also been found to be less efficient than self-consistency approaches (Huang et al., 2024). These issues and limitations raise questions about the true intrinsic self-correcting capabilities of LLMs (Huang et al., 2024). In contrast with those methods, our approach does not involve oracle labels and feedback from external verifiers. Our approach completely depends on the model’s intrinsic self-feedback ability.

Consistency in Reasoning Steps Numerous types of research showcase that the accuracy of the final answer is influenced by the consistency of reasoning steps (Wang et al., 2023c; Li et al., 2023c). These approaches typically involve a “oversample-then-select” framework (Shridhar et al., 2023; Cobbe et al., 2021b; Weng et al., 2023), where methods like self-consistency sample multiple reasoning steps (Wang et al., 2023c) and then select the most consistent or reliable response, e.g. self-consistency samples the reasoning steps many times (Wang et al., 2023c), Adaptive Consistency which reduces sampling to 7.9 times with an early stop criterion, and SCREWS (Shridhar et al., 2023)

162 which integrates multiple selection methods like majority-voting and machine-selection. Confidence
 163 Matters (Li et al., 2024a) and Think Twice (Li et al., 2024b) sample answers and prompt the model
 164 to generate a new answer if conflicts arise between the first two responses. While all current works
 165 focus on the consistency on the reasoning steps, our method is crafted to focus on the consistency
 166 between feedback.

167
 168
 169 **Scaling Inference-Time Compute** Recent studies have explored scaling inference-time compute
 170 to improve LLM performance (Brown et al., 2024; Snell et al., 2024). These approaches focus on
 171 increasing the number of answers or optimizing compute allocation to enhance output quality. In
 172 contrast, our approach introduces a new dimension by scaling feedback, not just answer outputs. This
 173 shift from sampling more answers to refining the quality and consistency of feedback expands the
 174 possibilities for improving model accuracy.
 175

176
 177 **Feedback Quality Evaluation** Recent studies focus on evaluating the quality of feedback to
 178 enhance the self-correction ability of LLMs (Sun et al., 2024). Evaluating how well LLMs’ out-
 179 puts adhere to human values and ethical standards involves assessing biases, toxicity, and truthful-
 180 ness (Hendrycks et al., 2023; Huang et al., 2023). Various approaches utilize both LLMs (OpenAI
 181 et al., 2024; Fu et al., 2023; Liu et al., 2023; Ke et al., 2023; Li et al., 2023b) and humans (Saunders
 182 et al., 2022; Wang et al., 2023b) as critics or annotators to evaluate and improve generated outputs.
 183 CriticBench (Lan et al., 2024) introduces a benchmark for assessing feedback and meta-feedback
 184 capabilities, emphasizing complex reasoning tasks and demonstrating that meta-feedback can sig-
 185 nificantly impact downstream performance. Recent works have proposed benchmarks to evaluate
 186 LLMs’ ability to assess outputs: LLMBAR (Zeng et al., 2024) focuses on instruction-following, while
 187 ReaLMistake (Kamoi et al., 2024a) evaluates error detection across multiple categories. In contrast
 188 to the recent works, our method involves a analysis of feedback to detect and correct inconsistencies
 189 across multiple sampling, focusing on refining feedback before revising the original response. This
 190 approach provides a more targeted enhancement in feedback quality compared to existing benchmarks
 191 that center on instruction-following and error detection.
 192
 193
 194

195 3 METHOD

196
 197
 198 In this section, we introduce the **Feedback-on-Feedback (FoF)** prompting method, which follows a
 199 three-step feedback refinement process: feedback generation, meta-feedback generation, and feedback
 200 refinement. A detailed FoF algorithm can be found in Appendix G. In this section, we first introduce
 201 the feedback generation and meta-feedback generation steps, and then we introduce the feedback
 202 refinement process.
 203
 204

205 **Base Generator** The base generator is an LLM that takes the question Q as input and generates
 206 an initial answer R_0 . The initial answer is generated using zero-shot chain-of-thought prompting
 207 (Brown et al., 2020; Wei et al., 2023). Following (Madaan et al., 2023), we use the same generation
 208 prompt p_{gen} . Given an input question Q , a generation prompt p_{gen} , and a base generator BG , the
 209 initial answer R_0 is generated based on the combination of p_{gen} and Q .
 210
 211

212
 213 **Critic Model** The critic model is another LLM that takes the CoT which contains the initial answer
 214 R_0 and the question Q as input and provides feedback on the quality of the answer. To generate the
 215 feedback, we prompt the critic model with the prompt p_{fb} . All the prompts used for the critic are
 shown in the textbox below.

216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269

Feedback Prompt:
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 reasoning steps and check if everything looks good.

FoF Prompt:
Disagree: Here are the two sampling feedback from the critic model on your previously generated reasoning step: Feedback sample 1: xxx. Feedback sample 2: xxx. The critic model is giving two different types of feedback, check the feedback and give the best feedback
Need Clarification = Here are the two sampling feedback...Clarifications are needed from the sampling feedback, try to clarify the feedback.

Refined Feedback Prompt: The programmer model may need clarification or disagree with you: FoF: xxx. Please give only one refined feedback based on the FoF from the programmer model. Your response should be similar to the previous round of feedback.

The critic model samples feedback with a temperature of 0.7 to generate F_1 and F_2 based on its training data and the given prompt.

Feedback Refinement The feedback refinement process aims to improve feedback quality and generate refined answers. It consists of the following steps:

FEEDBACK SIMILARITY CHECK We compute the semantic similarity S between two feedback samples F_1 and F_2 using cosine similarity. We utilize the TF-IDF vectorization method (Jones, 2021) to transform the feedback samples into vectors. First, we apply TF-IDF to convert F_1 and F_2 into numerical representations, denoted as $F1_vector$ and $F2_vector$. Following the vectorization, we calculate the cosine similarity between these two vectors using the formula:

$$S = \frac{F1_vector \cdot F2_vector}{\|F1_vector\| \|F2_vector\|} \quad (1)$$

This allows us to quantify the similarity between the two feedback samples based on their vector representations. The semantic similarity thresholds θ_1 and θ_2 , are set at 0.5 and 0.8 respectively. These thresholds were chosen based on manual inspection of a few examples from the validation set. Based on these thresholds, we categorize the feedback similarity levels as follows:

- If $0 \leq S < \theta_1$, the feedback samples are considered to disagree with each other.
- If $\theta_1 \leq S \leq \theta_2$, the feedback samples need clarification, examples could be found in Section 5.3.
- If $S > \theta_2$, the feedback samples are considered to agree with each other.

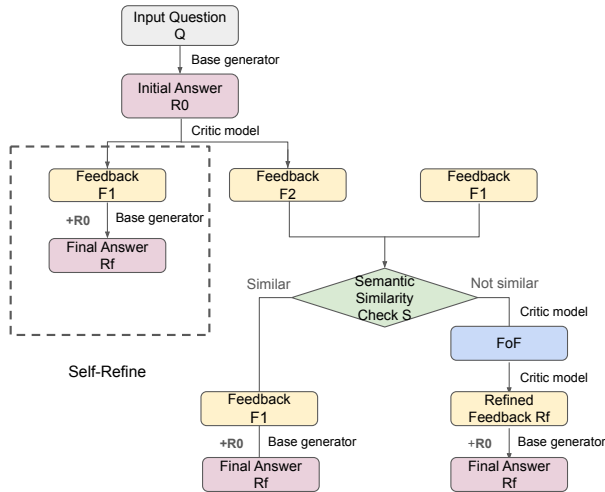


Figure 2: Block Diagram of the FoF method and the Self-Refine baseline. The left-hand side illustrates the Self-Refine method (Madaan et al., 2023), where the model generates feedback on its initial answer R_0 , refines the answer using the feedback and R_0 , and produces a final refined answer R_f . The right-hand side demonstrates the FoF method, which samples two feedback responses F_1 and F_2 , comparing them through a semantic similarity check, and generating a foF if inconsistencies are found. The refined answer R_f is generated based on the aggregated feedback and R_0 .

270 FEEDBACK-ON-FEEDBACK (FOF) GENERATION If the feedback samples F_1 and F_2 have low
 271 similarity, we generate FoF using the base generator BG and the prompt p_{fof} .
 272

273 REFINED FEEDBACK GENERATION The refined feedback RF is generated by the critic model
 274 CM using all the history contexts including the question Q , the initial answer R_0 , the FoF, and the
 275 feedback samples F_1 and F_2 , and the prompt p_{rf} .
 276

277 **Final Answer Refinement** The final refined answer R_f is generated by the base generator BG
 278 using the question Q , the initial answer R_0 , and the refined feedback RF , along with the refined
 279 answer prompt p_{ra} .

280 The refined answer R_f is the final output of the FoF prompting method, which incorporates the
 281 feedback and refinement process to improve the accuracy and reliability of the generated answer.
 282

283 4 EXPERIMENTS

284
 285 **Models** We utilize state-of-the-art language models as the base generator and critic in our FoF
 286 framework. We evaluate two LLMs, GPT-3.5-turbo and LLaMA3-8B, as our base models. We utilize
 287 the LLaMA3-8B model, which balances advanced capabilities with computational efficiency. Since
 288 GPT4 is considered as a strong model due to its performance on various benchmarks (OpenAI et al.,
 289 2024), we show the usage of GPT-4 as critic, showing that higher-quality feedback from a strong
 290 model can enhance accuracy, without any additional model training.
 291

292 **Benchmarks** We evaluate the performance of our FoF approach on three benchmarks requiring
 293 various reasoning skills. These evaluations span multiple types of tasks, covering arithmetic reasoning,
 294 commonsense reasoning, and programming problem-solving:
 295

296 **MATH REASONING:** We use the GSM8K dataset (Cobbe et al., 2021a), comprising 8.5K grade
 297 school math word problems to assess multi-step reasoning and numerical accuracy. For our evaluation,
 298 we specifically utilize the test set from GSM8K, which contains 1,319 examples.

299 **MACHINE TRANSLATION:** We employ the Commonsense Machine Translation (CSMT) dataset
 300 (He et al., 2020) to evaluate translation quality using automatic metrics BLEURT (Sellam et al., 2020)
 301 and COMET (Stewart et al., 2020). BLEURT is a learned evaluation metric based on BERT, focusing
 302 on fluency and the extent to which the candidate conveys the meaning of the reference. COMET, on
 303 the other hand, is a neural framework that uses source text along with gold translations to measure
 304 both fluency and semantic accuracy. We take the test set from CSMT, which contains 200 examples.
 305

306 **PROGRAMMING PROBLEM SOLVING:** We use the MBPP (Multiple Benchmark Programming
 307 Problems) dataset (Austin et al., 2021), featuring 974 Python problems to test the model’s ability
 308 to generate correct code given task description as input. We perform experiments on the test set of
 309 MBPP, which contains 500 python problems, where each problem has 3 unit tests. We follow prior
 310 work in including the first unit test in the prompt as part of the problem description (Chen et al.,
 311 2023b; 2021), and keep the remaining 2 unit tests hidden for a full evaluation. We evaluate MBPP
 312 based on the pass@1 metric, which indicates whether the single generated solution is correct (Chen
 313 et al., 2021).

314 **Prompt Selection Process** Since LLMs are known to be sensitive to different prompts (Huang et al.,
 315 2024; Li et al., 2024a), to evaluate the impact of different feedback prompts on model performance,
 316 we experiment with several prompts inspired by related works (Huang et al., 2024). Appendix B
 317 presents the results of FoF using various prompts for the test sets of GSM8K and MBPP datasets.
 318 We found that the variance between prompts did not significantly affect the final results, as scores
 319 for the GPT-3.5-0515 model were relatively consistent, ranging from 74.22 to 79.22, and for the
 320 LLaMA-3-8B model, the scores range from 45.17 to 46.92, indicating some variability but not
 321 a drastic impact on overall performance. We use the same prompt for both FoF and Self-Refine
 322 (Madaan et al., 2023) to ensure a fair comparison.
 323

Baselines This section provides an overview of the baseline methods, including:

CO_T-PROMPTING: Chain-of-Thought (CoT) prompting (Wei et al., 2023) is a technique that elicits reasoning in large language models by encouraging them to generate intermediate reasoning steps before arriving at the final answer. This method enhances the model’s ability to solve complex problems by breaking down the problem-solving process into smaller steps. The prompt typically contains instructions such as “let’s think step by step”.

SELF-REFINE PROMPTING: The primary baseline method in this study is the Self-Refine method (Madaan et al., 2023). Self-refine prompting is an iterative refinement method where the model generates self-feedback and uses it to improve its initial outputs. Huang et al. (2024) refer to this as critical prompting, which includes instructions like “find the error in your reasoning step”. To ensure fairness between the Self-Refine and FoF settings, both methods start with the same initial answer and feedback round.

4.1 FEEDBACK SAMPLING

In our experiments, since we need to sample multiple feedback to generate meta-feedback, we sample two feedback responses from the critic model due to the context limit of GPT-3.5-Turbo with a temperature of 0.7. This temperature value ensures that the generated feedback samples are diverse (Renze & Guven, 2024; Wang et al., 2020; 2023a), allowing us to test the core idea of generating meta-feedback effectively.

4.2 STOP CONDITION

We follow the setup by Self-Refine (Madaan et al., 2023), where the feedback refinement process stops when it reaches the feedback round limit, or when the feedback contains the phrase “there is no error”.

5 RESULT

Table 1: Performance comparison of different feedback methods across various models and datasets. Results are averaged over 3 runs with temperature=0.7, maintaining feedback randomness.

		GSM8K		CSMT		MBPP
		Acc	Oracle Acc	BLEURT	COMET	Acc
GPT-3.5-0515	+ Initial Answer	77.9±1.3	77.9±1.8	63.8±3.1	71.5±1.5	71.5±2.7
	+ Self-Consistency@10	78.3±2.4	-	-	-	74.5±1.6
	+ Self-refine	77.4±1.7	78.8±1.9	66.1±0.8	74.1±2.3	74.1±0.5
	+ FoF	78.7±2.0	80.1±1.2	67.4±2.1	75.3±1.7	75.3±2.3
Llama-3-8B	+ Initial Answer	41.6±1.4	41.6±2.5	60.3±1.1	62.5±2.0	45.3±1.7
	+ Self-Consistency@10	42.2±3.2	-	-	-	45.4±1.9
	+ Self-refine	43.5±1.0	44.0±2.8	63.1±1.7	66.0±2.4	49.1±0.9
	+ FoF	45.2±1.9	45.7±1.4	66.3±2.7	68.0±3.1	51.7±2.4

5.1 MAIN RESULT

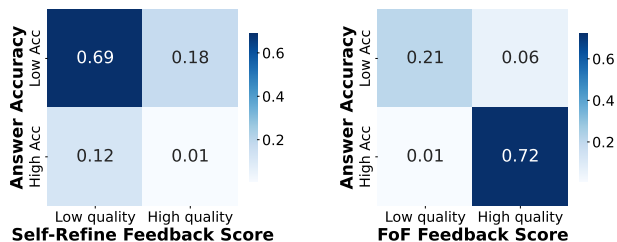
We perform evaluations using two different large-scale models across three benchmark datasets. As summarized in Table 1, the FoF method consistently demonstrates improvements across all benchmarks compared to the standard prompt and Self-Refine. For instance, using GPT-3.5-0515, our FoF method achieved an average accuracy of 78.71% on GSM8K, representing a 0.79% improvement over the standard prompt and a slight increase compared to Self-Refine. It is notable that the performance of GPT-3.5-0515 on GSM8K decreases after applying Self-Refine, this is aligned with the finding of (Huang et al., 2024). In some cases, Self-Refine even led to a decline in accuracy due to errors in the feedback. Our method addresses this issue by enhancing the quality of feedback through meta-feedback, which subsequently improves the final accuracy. Notably, for the LLaMA3-8B model, the FoF method achieve 45.17% accuracy, marking a 3.58% improvement over the standard prompt and a 1.68% increase compared to Self-Refine. The improvements from our method tend to decrease

378 as the model capability increases, yet the decision refinement stage consistently enhances performance
 379 across all models. It is notable that the performance of GPT-3.5-0515 on GSM8K decreases after
 380 applying Self-Refine, this is aligned with the finding of Huang et al. (2024). In the MBPP task, we
 381 assessed the effectiveness of the FoF method using the GPT-3.5-0515 and LLaMA3-8B models. As
 382 shown in Table 1, the FoF method achieved an accuracy of 75.27% with GPT-3.5-0515, reflecting
 383 a 3.77% improvement over the standard prompt and a 1.19% increase compared to the Self-Refine
 384 method.

385 In the Machine Translation Tasks, we evaluate the performance using the BLEU and COMET metrics.
 386 Our FoF approach achieves significant improvements in both BLEU and COMET scores after 4
 387 rounds of iterative refinement. The BLEU score increases from 63.77 to 67.37, while the COMET
 388 score improves from 71.5 to 75.27. These results demonstrate the effectiveness of the FoF mechanism
 389 in enhancing the quality of the generated translations via iterative feedback and refinement rounds.

391 **Higher Feedback Quality Leads to**

392 **Better Answers** Figure 3 visual-
 393 izes the correlation between feedback
 394 scores, provided by the GPT-3.5-0515
 395 LLM judge and CriticBench prompts
 396 (Lan et al., 2024), and answer accu-
 397 racies for both methods. The FoF
 398 heatmap reveals a strong positive cor-
 399 relation, with 72% of data points
 400 falling into the high feedback score
 401 and high answer accuracy quadrant.
 402 In contrast, the Self-Refine heatmap
 403 shows a weaker correlation, with data
 404 points more evenly distributed across
 all quadrants.



(a) Self-Refine Approach. 42% of data falls in the low feedback and low accuracy quadrant, indicating weak correlation between feedback quality and answer accuracy.
 (b) FoF Approach. 72% of data falls in the high feedback and high accuracy quadrant, demonstrating a strong positive correlation.

405 Our analysis suggests that refining
 406 feedback through an iterative pro-
 407 cess improves the correlation between
 408 feedback scores and answer accuracy.
 409 This finding is consistent with Crit-
 410 icBench (Lan et al., 2024), which
 411 states that higher feedback quality
 412 leads to improved accuracy in ques-
 413 tion answering.

Figure 3: Heatmaps comparing the correlation between feed-
 back score, which is prompted and calculated by the LLM,
 and answer accuracy for Self-Refine and FoF approaches.
 Note the imbalance: 228/500 examples have Self-Refine
 feedback, while 118/500 have FoF feedback. Despite fewer
 examples for FoF, higher feedback quality leads to higher
 accuracy in the MBPP task.

414 **FoF Changes More Answers Than Self-Refine** We further evaluate the changes in the answers
 415 after applying self-correction with the FoF method. The results on the GSM8K dataset using the
 416 GPT-3.5-0515 model show that our FoF method significantly increases the rate of Incorrect →
 417 Correct changes, demonstrating its effectiveness in enhancing answer accuracy. While both methods
 418 have similar percentages of wrong-to-wrong transitions (22.5% for FoF and 22.4% for Self-Refine),
 419 FoF outperforms Self-Refine in the wrong-to-correct category (3.2% vs. 3.5%) in each round. FoF
 420 generates more diverse answers than Self-Refine (Madaan et al., 2023) due to the additional meta-
 421 feedback stage, which encourages variability in response generation. This aligns with Huang et al.
 422 (2024), who note that mischanges from correct answer to incorrect result in self-correction failures.
 423 The improvements of FoF across tasks are due to fewer mischanges in feedback and answer rounds.

424 **Comparison with Self-Consistency** To ensure a fair comparison, we used a similar total number of
 425 tokens during inference between our FoF method and the self-consistency approach. Self-consistency
 426 involves generating 10 samples per iteration, while FoF involves one initial answer, three rounds of
 427 generation, two sampled feedback, one meta-feedback, and one refined answer, totaling 16 inference
 428 steps. Our results (Table 1) show that FoF consistently outperforms self-consistency across both
 429 GSM8K and MBPP datasets, with accuracy improvements ranging from 0.5% to 3%.

431 5.2 ABLATION STUDIES

Critic Quality Matters We conduct an ablation study to investigate the impact of the critic model’s quality on the final performance of our FoF approach. We compared two critic models, GPT-3.5 and GPT-4, while keeping the base generator fixed as GPT-3.5. Table 2 presents the results on the GSM8K dataset. The findings highlight the importance of the critic model’s quality in the FoF framework. By employing a more advanced language model as the critic, the system can generate higher-quality critiques, which in turn guide the base generator to produce more accurate corrections.

Table 2: Ablation study on the impact of critic model quality on final accuracy. Results are shown for the GSM8K dataset with GPT-3.5 as the base generator and using GPT-3.5, GPT-4 as the critic model.

Base Model	Critic Model	Prompt Type	# of Feedback Samples	GSM8K
				Accuracy
GPT-3.5	GPT-3.5	+ Standard Prompt	0	77.27
		+ Self-refine	0	79.26
		+ Self-refine	2	77.78
		+ FoF	2	79.79
GPT-3.5	GPT-4	+ Standard Prompt	0	78.24
		+ Self-refine	0	85.88
		+ Self-refine	2	85.48
		+ FoF	2	86.05

Feedback Sampling Consistency

We also include a self-refine with two sampling variants to ensure a comparison using the same amount of API calls and a similar number of tokens. We sampled two feedbacks to not exceed the token limit of 4096. In this setting, self-refine generates two sampling feedback, and the base generator selects the one it has the most confidence in by using a prompt “Please compare the two pieces of feedback and choose the most appropriate one as the final feedback”. The results indicate that LLMs lack the ability to choose the best feedback without external verification mechanisms, such as consistency checks. We further analyze the importance of feedback consistency by introducing a new baseline, USC on Feedback, which combines the Self-Refine approach (Madaan et al., 2023) with Universal Self-Consistency (USC) methods (Chen et al., 2023a). As highlighted in the appendix (Section A, the FoF method outperforms USC on Feedback by improving the accuracy on the GSM-Plus dataset (Li et al., 2024c) from 70% to 75%. The takeaway from these results is that incorporating consistency checks and leveraging multiple feedback samples can significantly enhance the performance of self-refinement methods in LLMs.

5.3 CASE STUDY

Failed Examples and Analysis While our FoF method demonstrates significant improvements in self-correction capabilities, it is important to acknowledge and analyze the instances where the method did not perform as expected. In this example, the initial answer is correct but includes incorrect intermediate steps. The two feedback samples provide incorrect guidance, leading FoF to agree with the inaccurate feedback and produce a refined response that reinforces the wrong answer. This case illustrates how multiple inaccurate feedback samples can compound errors, distracting FoF from identifying the correct solution. Consequently, FoF fails to recognize the initial answer as correct and follows the misleading feedback, resulting in an incorrect final answer. More failed and successful examples are available in Appendix D.5 and D.6.

Gold Answer: (...detailed reasoning steps) $7*2=14$

Initial Answer: (...detailed reasoning steps) $5*(14/5) = 5*2.8 = 14$ (Correct initial answer with wrong middle steps)

Feedback sample 1: The solution should round $14/5$ (which equals 2.8) up to 3. (Wrong feedback which focuses on fractions.)

Feedback sample 2: There are 7 people in total so we should use $7*2 = 14$. (Correct feedback which leads to the correct answer) **Clarifications are needed.**

FoF: The first feedback identifies the issue with dividing 14 by 5 and suggests rounding up, which results in 15. Based on the first feedback approach to the calculation, I agree that rounding up gives a more appropriate estimate. (FoF agrees with the incorrect feedback)

6 CONCLUSION AND FUTURE WORK

In this study, we investigate the FoF approach for enhancing the intrinsic self-correction (Huang et al., 2024) capability of LLMs by applying meta-feedback (Lan et al., 2024) in the feedback refinement process. Using our FoF method, LLMs achieve higher accuracy in math reasoning tasks and better quality in generation tasks and machine translation tasks. The study shows that the quality of the critic model is crucial in the FoF framework, as higher-quality critics generate better feedback, which positively correlates with improved performance and accuracy of the base generator.

In this work, we only explore sampling two pieces of feedback. Future work could extend this to multiple pieces of feedback to further enhance the feedback refinement process. Future work could explore integrating a reward mechanism into different stages of self-correction, such as reasoning steps, feedback, and meta-feedback, to guide the self-correction process more effectively (Yuan et al., 2024). Introducing a self-rewarding model that updates rewards during training could potentially overcome the limitations of treating all feedback equally and improve alignment with desired outcomes. Moreover, techniques such as multi-agent reasoning (Haji et al., 2024) and Constrained Chain-of-ToM (CCoToM) prompting (Lin et al., 2024) could further enhance the model’s ability to understand and predict nuanced human intentions. Incorporating Logic-of-Thought (LoT) (Liu et al., 2024) to maintain logical consistency and integrating human-in-the-loop mechanisms (Cai et al., 2023) could refine the feedback process and improve model performance across diverse scenarios.

REFERENCES

- Pranjal Aggarwal, Aman Madaan, Yiming Yang, and Mausam. Let’s sample step by step: Adaptive-consistency for efficient reasoning and coding with llms, 2023.
- Afra Feyza Akyurek, Ekin Akyurek, Ashwin Kalyan, Peter Clark, Derry Tanti Wijaya, and Niket Tandon. RL4F: Generating natural language feedback with reinforcement learning for repairing model outputs. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 7716–7733, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.427. URL <https://aclanthology.org/2023.acl-long.427>.
- Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models, 2021.
- Bradley Brown, Jordan Juravsky, Ryan Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. Large language monkeys: Scaling inference compute with repeated sampling, 2024. URL <https://arxiv.org/abs/2407.21787>.
- Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020.
- Zefan Cai, Baobao Chang, and Wenjuan Han. Human-in-the-loop through chain-of-thought, 2023.
- Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021.

- 540 Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash,
541 Charles Sutton, Xuezhi Wang, and Denny Zhou. Universal self-consistency for large language
542 model generation, 2023a. URL <https://arxiv.org/abs/2311.17311>.
- 543 Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. Teaching large language models to
544 self-debug, 2023b.
- 545 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
546 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
547 Schulman. Training verifiers to solve math word problems, 2021a.
- 548 Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser,
549 Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John
550 Schulman. Training verifiers to solve math word problems, 2021b. URL <https://arxiv.org/abs/2110.14168>.
- 551 Roi Cohen, May Hamri, Mor Geva, and Amir Globerson. Lm vs lm: Detecting factual errors via
552 cross examination, 2023.
- 553 Yilun Du, Shuang Li, Antonio Torralba, Joshua B. Tenenbaum, and Igor Mordatch. Improving
554 factuality and reasoning in language models through multiagent debate, 2023.
- 555 Jinlan Fu, See-Kiong Ng, Zhengbao Jiang, and Pengfei Liu. Gptscore: Evaluate as you desire, 2023.
- 556 Zelalem Gero, Chandan Singh, Hao Cheng, Tristan Naumann, Michel Galley, Jianfeng Gao, and
557 Hoifung Poon. Self-verification improves few-shot clinical information extraction, 2023.
- 558 Zhibin Gou, Zhihong Shao, Yeyun Gong, Yelong Shen, Yujiu Yang, Nan Duan, and Weizhu Chen.
559 Critic: Large language models can self-correct with tool-interactive critiquing, 2024.
- 560 Fatemeh Haji, Mazal Bethany, Maryam Tabar, Jason Chiang, Anthony Rios, and Peyman Najafirad.
561 Improving llm reasoning with multi-agent tree-of-thought validator agent, 2024. URL <https://arxiv.org/abs/2409.11527>.
- 562 Jie He, Tao Wang, Deyi Xiong, and Qun Liu. The box is in the pen: Evaluating commonsense
563 reasoning in neural machine translation. In Trevor Cohn, Yulan He, and Yang Liu (eds.), *Findings of*
564 *the Association for Computational Linguistics: EMNLP 2020*, pp. 3662–3672, Online, November
565 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.327.
566 URL <https://aclanthology.org/2020.findings-emnlp.327>.
- 567 Dan Hendrycks, Collin Burns, Steven Basart, Andrew Critch, Jerry Li, Dawn Song, and Jacob
568 Steinhardt. Aligning ai with shared human values, 2023.
- 569 Jie Huang, Xinyun Chen, Swaroop Mishra, Huaixiu Steven Zheng, Adams Wei Yu, Xinying Song,
570 and Denny Zhou. Large language models cannot self-correct reasoning yet, 2024.
- 571 Yue Huang, Qihui Zhang, Philip S. Y, and Lichao Sun. Trustgpt: A benchmark for trustworthy and
572 responsible large language models, 2023.
- 573 Karen Spärck Jones. A statistical interpretation of term specificity and its application in re-
574 trieval. *J. Documentation*, 60:493–502, 2021. URL <https://api.semanticscholar.org/CorpusID:2996187>.
- 575 Ryo Kamoi, Sarkar Snigdha Sarathi Das, Renze Lou, Jihyun Janice Ahn, Yilun Zhao, Xiaoxin Lu,
576 Nan Zhang, Yusen Zhang, Ranran Haoran Zhang, Sujeeth Reddy Vummanthala, Salika Dave,
577 Shaobo Qin, Arman Cohan, Wenpeng Yin, and Rui Zhang. Evaluating llms at detecting errors in
578 llm responses, 2024a. URL <https://arxiv.org/abs/2404.03602>.
- 579 Ryo Kamoi, Yusen Zhang, Nan Zhang, Jiawei Han, and Rui Zhang. When can llms actually
580 correct their own mistakes? a critical survey of self-correction of llms, 2024b. URL <https://arxiv.org/abs/2406.01297>.
- 581 Pei Ke, Bosi Wen, Zhuoer Feng, Xiao Liu, Xuanyu Lei, Jiale Cheng, Shengyuan Wang, Aohan Zeng,
582 Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. Critiquellm: Scaling llm-as-critic for
583 effective and explainable evaluation of large language model generation, 2023.

- 594 Geunwoo Kim, Pierre Baldi, and Stephen McAleer. Language models can solve computer tasks,
595 2023.
596
- 597 Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate
598 Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha
599 Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra
600 Faust. Training language models to self-correct via reinforcement learning, 2024. URL <https://arxiv.org/abs/2409.12917>.
601
- 602 Tian Lan, Wenwei Zhang, Chen Xu, Heyan Huang, Dahua Lin, Kai Chen, and Xian ling Mao.
603 Criticbench: Evaluating large language models as critic, 2024.
604
- 605 Loka Li, Guangyi Chen, Yusheng Su, Zhenhao Chen, Yixuan Zhang, Eric Xing, and Kun Zhang.
606 Confidence matters: Revisiting intrinsic self-correction capabilities of large language models,
607 2024a.
608
- 609 Moxin Li, Wenjie Wang, Fuli Feng, Fengbin Zhu, Qifan Wang, and Tat-Seng Chua. Think twice
610 before assure: Confidence estimation for large language models through reflection on multiple
611 answers, 2024b.
- 612 Qintong Li, Leyang Cui, Xueliang Zhao, Lingpeng Kong, and Wei Bi. Gsm-plus: A comprehensive
613 benchmark for evaluating the robustness of llms as mathematical problem solvers, 2024c. URL
614 <https://arxiv.org/abs/2402.19255>.
615
- 616 Ruosen Li, Teerth Patel, and Xinya Du. Prd: Peer rank and discussion improve large language model
617 based evaluations, 2023a.
- 618 Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy
619 Liang, and Tatsunori B. Hashimoto. AlpacaEval: An automatic evaluator of instruction-following
620 models. https://github.com/tatsu-lab/alpaca_eval, 2023b.
621
- 622 Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. Making
623 large language models better reasoners with step-aware verifier, 2023c.
- 624 Tian Liang, Zhiwei He, Wenxiang Jiao, Xing Wang, Yan Wang, Rui Wang, Yujiu Yang, Zhaopeng Tu,
625 and Shuming Shi. Encouraging divergent thinking in large language models through multi-agent
626 debate, 2023a.
627
- 628 Weixin Liang, Yuhui Zhang, Hancheng Cao, Binglu Wang, Daisy Ding, Xinyu Yang, Kailas Vodra-
629 halli, Siyu He, Daniel Smith, Yian Yin, Daniel McFarland, and James Zou. Can large language
630 models provide useful feedback on research papers? a large-scale empirical analysis, 2023b.
631
- 632 Zizheng Lin, Chunkit Chan, Yangqiu Song, and Xin Liu. Constrained reasoning chains for enhancing
633 theory-of-mind in large language models, 2024. URL <https://arxiv.org/abs/2409.13490>.
634
- 635 Tongxuan Liu, Wenjiang Xu, Weizhe Huang, Xingyu Wang, Jiaying Wang, Hailong Yang, and Jing
636 Li. Logic-of-thought: Injecting logic into contexts for full reasoning in large language models,
637 2024. URL <https://arxiv.org/abs/2409.17539>.
638
- 639 Yang Liu, Dan Iter, Yichong Xu, Shuohang Wang, Ruochen Xu, and Chenguang Zhu. G-eval: NLG
640 evaluation using gpt-4 with better human alignment. In Houda Bouamor, Juan Pino, and Kalika
641 Bali (eds.), *Proceedings of the 2023 Conference on Empirical Methods in Natural Language
642 Processing*, pp. 2511–2522, Singapore, December 2023. Association for Computational Linguistics.
643 doi: 10.18653/v1/2023.emnlp-main.153. URL [https://aclanthology.org/2023.
644 emnlp-main.153](https://aclanthology.org/2023.emnlp-main.153).
- 645 Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegrefe, Uri Alon,
646 Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder,
647 Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. Self-refine: Iterative
refinement with self-feedback, 2023.

- 648 OpenAI. Gpt-4o-mini: Advancing cost-efficient intelligence, 2024. URL [https://openai.com/](https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/)
649 [index/gpt-4o-mini-advancing-cost-efficient-intelligence/](https://openai.com/index/gpt-4o-mini-advancing-cost-efficient-intelligence/). Accessed:
650 October 1, 2024.
- 651
- 652 OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni
653 Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor
654 Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian,
655 Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny
656 Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks,
657 Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea
658 Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen,
659 Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung,
660 Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch,
661 Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty
662 Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte,
663 Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel
664 Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua
665 Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike
666 Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon
667 Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne
668 Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo
669 Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar,
670 Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik
671 Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich,
672 Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy
673 Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie
674 Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini,
675 Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne,
676 Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David
677 Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie
678 Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély,
679 Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo
680 Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano,
681 Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng,
682 Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto,
683 Michael, Pokorný, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power,
684 Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis
685 Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted
686 Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel
687 Shams, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon
688 Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky,
689 Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang,
690 Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston
691 Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya,
692 Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason
693 Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff,
694 Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu,
695 Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba,
696 Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang,
697 William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024.
- 698 Liangming Pan, Michael Saxon, Wenda Xu, Deepak Nathani, Xinyi Wang, and William Yang Wang.
699 Automatically correcting large language models: Surveying the landscape of diverse self-correction
700 strategies, 2023.
- 701 Debjit Paul, Mete Ismayilzada, Maxime Peyrard, Beatriz Borges, Antoine Bosselut, Robert West, and
Boi Faltings. Refiner: Reasoning feedback on intermediate representations, 2024.
- Matthew Renze and Erhan Guven. The effect of sampling temperature on problem solving in large
language models, 2024.

- 702 William Saunders, Catherine Yeh, Jeff Wu, Steven Bills, Long Ouyang, Jonathan Ward, and Jan
703 Leike. Self-critiquing models for assisting human evaluators, 2022.
704
- 705 Thibault Sellam, Dipanjan Das, and Ankur Parikh. BLEURT: Learning robust metrics for text
706 generation. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel Tetreault (eds.), *Proceedings of*
707 *the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7881–7892, Online,
708 July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.704. URL
709 <https://aclanthology.org/2020.acl-main.704>.
- 710 Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu
711 Yao. Reflexion: Language agents with verbal reinforcement learning, 2023.
712
- 713 Kumar Shridhar, Harsh Jhamtani, Hao Fang, Benjamin Van Durme, Jason Eisner, and Patrick Xia.
714 Screws: A modular framework for reasoning with revisions, 2023.
715
- 716 Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally
717 can be more effective than scaling model parameters, 2024. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2408.03314)
718 [2408.03314](https://arxiv.org/abs/2408.03314).
- 719 Craig Stewart, Ricardo Rei, Catarina Farinha, and Alon Lavie. COMET - deploying a new state-
720 of-the-art MT evaluation metric in production. In Janice Campbell, Dmitriy Genzel, Ben Huyck,
721 and Patricia O’Neill-Brown (eds.), *Proceedings of the 14th Conference of the Association for*
722 *Machine Translation in the Americas (Volume 2: User Track)*, pp. 78–109, Virtual, October 2020.
723 Association for Machine Translation in the Americas. URL [https://aclanthology.org/](https://aclanthology.org/2020.amta-user.4)
724 [2020.amta-user.4](https://aclanthology.org/2020.amta-user.4).
- 725 Shichao Sun, Junlong Li, Weizhe Yuan, Ruifeng Yuan, Wenjie Li, and Pengfei Liu. The critique of
726 critique, 2024.
727
- 728 Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée
729 Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand
730 Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language
731 models, 2023.
732
- 733 Karthik Valmeekam, Matthew Marquez, and Subbarao Kambhampati. Can large language models
734 really improve by self-critiquing their own plans?, 2023. URL [https://arxiv.org/abs/](https://arxiv.org/abs/2310.08118)
735 [2310.08118](https://arxiv.org/abs/2310.08118).
- 736 Chi Wang, Susan Xueqing Liu, and Ahmed H. Awadallah. Cost-effective hyperparameter optimization
737 for large language model generation inference, 2023a.
738
- 739 Pei-Hsin Wang, Sheng-Iou Hsieh, Shih-Chieh Chang, Yu-Ting Chen, Jia-Yu Pan, Wei Wei, and
740 Da-Chang Juan. Contextual temperature for language modeling, 2020.
741
- 742 Tianlu Wang, Ping Yu, Xiaoqing Ellen Tan, Sean O’Brien, Ramakanth Pasunuru, Jane Dwivedi-Yu,
743 Olga Golovneva, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. Shepherd: A
744 critic for language model generation, 2023b.
- 745 Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdh-
746 ery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models,
747 2023c.
748
- 749 Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le,
750 and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023.
- 751 Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin
752 Choi. Generating sequences by learning to self-correct, 2022.
753
- 754 Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu,
755 and Jun Zhao. Large language models are better reasoners with self-verification, 2023. URL
<https://arxiv.org/abs/2212.09561>.

756 Seonghyeon Ye, Yongrae Jo, Doyoung Kim, Sungdong Kim, Hyeonbin Hwang, and Minjoon Seo.
757 Selfee: Iterative self-revising llm empowered by self-feedback generation. Blog post, May 2023.
758 URL <https://kaistai.github.io/SelFee/>.
759

760 Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu,
761 and Jason Weston. Self-rewarding language models, 2024. URL [https://arxiv.org/abs/
762 2401.10020](https://arxiv.org/abs/2401.10020).

763 Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. Star: Bootstrapping reasoning with
764 reasoning, 2022.
765

766 Zhiyuan Zeng, Jiatong Yu, Tianyu Gao, Yu Meng, Tanya Goyal, and Danqi Chen. Evaluating large
767 language models at evaluating instruction following, 2024. URL [https://arxiv.org/abs/
768 2310.07641](https://arxiv.org/abs/2310.07641).

769 Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee,
770 Honglak Lee, and Lu Wang. Small language models need strong verifiers to self-correct rea-
771 soning, 2024. URL <https://arxiv.org/abs/2404.17140>.

772 Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang,
773 Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica.
774 Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809

A EXTRA CONSISTENCY EXPERIMENT

Recent advancements have introduced the universal self-consistency (USC) method (Chen et al., 2023a), in which LLMs are prompted to select the most consistent response from multiple generated answers. To further emphasize the importance of feedback consistency, we introduce a new baseline, **USC on Feedback**, combining the Self-Refine approach (Madaan et al., 2023) with USC (Chen et al., 2023a). In this baseline, we first sample N pieces of feedback and use the USC prompt to let LLMs select the most consistent feedback. This feedback is then used to refine the final answer. In order to make a fair comparison, we modify the FoF framework to operate under the same conditions as the USC on Feedback baseline. After generating N pieces of feedback, LLMs in the FoF approach identify inconsistencies and categorize feedback into three groups: Agree, Need Clarification, and Disagree. Based on this categorization, the framework proceeds with the usual FoF steps—generating refined feedback from the categorized responses, which is subsequently used to refine the answer. This experiment is conducted on the GSM-Plus dataset (Li et al., 2024c), using the cost-efficient and advanced GPT-4o-mini model (OpenAI, 2024). Since GPT-4o is trained based on GPT-4, and the GPT-4 training data includes GSM8K (Cobbe et al., 2021a), we opt to use GSM-Plus (Li et al., 2024c), an extended version of GSM8K that includes modifications such as numerical variation, arithmetic variation, problem rephrasing, distractor insertion, and critical thinking tasks. To maximize the use of the input token limit, we sample N=10 feedback in 1 round on 200 random shuffled example of GSM-Plus. As shown in Table 3, the FoF method improves the accuracy from 0.70 to 0.75 by selecting most consistent feedback.

Table 3: Results of GPT-4o-mini in GSMP-Plus

Method	Accuracy on GSM-Plus
Self-Consistency on Answer	0.76
USC on Feedback	0.70
FoF	0.75

B RESULTS OF PROMPT SELECTION

Table 4: Results of GPT-3.5-0515 and LLaMA-3-8B with different feedback prompts.

Feedback Prompt in FoF	GPT-3.5-0515		LLaMA-3-8B	
	GSM8K	MBPP	GSM8K	MBPP
Assume that this answer could be either correct or incorrect. Review the answer carefully and report any serious problems you find.	78.11	74.22	45.23	51.98
Review your previous answer and determine whether it’s correct. If wrong, find the problems with your answer.	78.79	74.49	46.09	52.53
Verify whether your answer is correct, and provide an explanation.	79.22	75.43	46.92	52.76
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 reasoning steps and check if everything looks good. (Our Prompt)	78.71	75.27	45.17	51.67

C COMPARISON BETWEEN FOF AND OTHER EXISTING WORKS

	Iterative Answer	Automated Critique	Zero-shot	Consistency on Answer	Consistency on Feedback
Self-Refine (Madaan et al., 2023)	✓	✓			
Self-Verification (Gero et al., 2023)		✓			
Reflexion (Shinn et al., 2023)	✓	✓			
Self-Debug (Chen et al., 2023b)		✓			
SelFee (Ye et al., 2023)	✓		✓		
Critical Prompt (Huang et al., 2024)	✓	✓	✓		
Think Twice (Li et al., 2024b)	✓	✓	✓	✓	
Confidence Matters (Li et al., 2024a)	✓	✓	✓	✓	
FoF (Ours)	✓	✓	✓	✓	✓

Table 5: Summary of related work on intrinsic self-correction (Huang et al., 2024) This table categorizes intrinsic self-correction methods based on five properties: Iterative Answer (the use of multiple rounds of answer generation), Automated Critique (whether the critique is generated automatically), Zero-shot (whether the method can perform without additional examples), Consistency on Answer (consistency in generating the correct answer), and Consistency on Feedback (consistency in the feedback provided).

D EXPERIMENT PROMPTS WITH DEMONSTRATION

Here are the shortened versions of the experimental prompts, including instructions and demonstrations. For clarity, only 1-2 demonstrations are provided per prompt. Each demonstrated result string is highlighted for readability, though this highlighting is not part of the actual prompt. Following each prompt, a test question and a cue (e.g., "Answer:") are provided to indicate where the result string should be placed.

D.1 INITIAL ANSWER GENERATION

FoF on GSM8K with the Initial Answer Round Shown Below

```
===== system =====
Can you answer this question by writing reasoning steps and the answer:
On Monday, Walt walked 4 miles. Tuesday, he walked 6 times as many miles as he walked
on Monday. His total mileage Monday through Wednesday was 41 miles. How many miles
did he walk on Wednesday?
Please ensure your reasoning steps only consists of the essential steps and the final answer
should be a single numerical number, in the form
boxedanswer, at the end of your response.
===== assistant =====
Reasoning steps: 1. On Monday, Walt walked 4 miles. 2. On Tuesday, he walked 6 times as
many miles as he walked on Monday. Therefore, he walked  $4 * 6 = 24$  miles on Tuesday. 3.
His total mileage Monday through Wednesday was 41 miles. So, he walked a total of  $4 + 24$ 
+  $x = 41$  miles, where  $x$  represents the number of miles he walked on Wednesday. 4. Solving
the equation, we get  $x = 41 - 4 - 24 = 13$ . Therefore, Walt walked 13 miles on Wednesday.
Answer: 13.
```

918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971

D.2 FEEDBACK GENERATION

FoF on GSM8K with 2 Sampled Feedback Shown Below

=====
assistant
=====
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 reasoning steps, and check if everything looks good

=====
user
=====
Here are the two sampling feedback from the critic model on your previously generated reasoning step:
The reasoning steps provided by the programmer model are incorrect. The mistake lies in step 4 where the equation is solved incorrectly. The correct equation should be $4 + 24 + x = 41$, not $4 + 24 + x = 41 - 4 - 24$. The reasoning steps provided by the programmer model are correct and logical. There are no logical issues with the steps to solve the math word problem. Clarifications are needed from the sampling feedback, try to clarify the feedback.

D.3 FoF GENERATION

FoF on GSM8K with Refined Feedback Shown Below

=====
assistant
=====
Based on the two sampling feedback, there seems to be a disagreement about the correctness of the reasoning steps provided by the programmer model. One feedback suggests that there is a mistake in step 4, while the other feedback states that the reasoning steps are correct and logical.

To clarify the feedback, it would be helpful to understand the specific issue pointed out in step 4 by the first feedback. Additionally, it would be beneficial to know whether there are any other concerns or suggestions related to the reasoning steps provided.

=====
user
=====
Here is the refined feedback from the critic model.
The reasoning steps provided by the programmer model are correct and logical. There is no mistake in step 4 where the equation is solved.

D.4 FINAL ANSWER GENERATION

FoF on GSM8K with the Final Answer Round Shown Below

=====
user
=====
Can you improve the reasoning steps in natural languages according to the feedback? Please ensure your final answer should be a single numerical number, in the form boxedanswer, at the end of your response.

=====
assistant
=====
Reasoning steps: 1. On Monday, Walt walked 4 miles. 2. On Tuesday, he walked 6 times as many miles as he walked on Monday. Therefore, he walked $4 * 6 = 24$ miles on Tuesday. 3. His total mileage Monday through Wednesday was 41 miles. So, the combined distance he walked on Monday, Tuesday, and Wednesday is $4 + 24 + x = 41$ miles, where x represents the number of miles he walked on Wednesday. 4. Solving the equation, we get $x = 41 - 4 - 24 = 13$.
Therefore, Walt walked 13 miles on Wednesday. Answer:
boxed13.

D.5 FAILED EXAMPLES AND ANALYSIS

While our Feedback-on-Feedback (FoF) method demonstrates significant improvements in self-correction capabilities, it is important to acknowledge and analyze the instances where the method

972 did not perform as expected. Understanding these failures can provide valuable insights into the
973 limitations of our approach and highlight areas for future improvement.
974

975 D.5.1 FAILED EXAMPLE 1: ARITHMETIC REASONING

976
977 In one instance from the GSM8K dataset, the initial answer model generated the following response
978 to a multi-step arithmetic problem:

979 Here are feedback from the critic model on your previously generated reasoning steps:

980
981 **Feedback 1:** The error in the reasoning steps is the assumption that the total
982 mileage for Monday through Wednesday is given as 41 miles. The prompt does not
983 provide any information about the total mileage for Monday through Wednesday.
984 Therefore, we cannot assume that the equation $7x + x + \text{Wednesday's mileage} = 41$
985 is correct. The correct approach would be to set up the equation $7x + \text{Wednesday's}$
986 $\text{mileage} = 41$, but we would need additional information or a different equation to
987 solve for Wednesday's mileage.

988 Despite iterative refinement, the final answer remained incorrect due to the conflicting nature of
989 the feedback provided. This indicates that our method struggles when presented with contradictory
990 critiques, especially in arithmetic reasoning tasks where precision is crucial.

991
992 Let's assume that Walt walked x miles on Monday. On Tuesday, he walked 6 times
993 as many miles as he walked on Monday, which is $6x$ miles. The total mileage for
994 Monday through Wednesday is given as 41 miles. Let's represent Wednesday's
995 mileage as y miles. So, the equation representing the total mileage would be $x + 6x$
996 $+ y = 41$. Simplifying the equation, we have $7x + y = 41$. Since we don't have any
997 other information, we cannot determine the exact values of x and y individually.
998 Therefore, we cannot determine the specific number of miles Walt walked on
999 Wednesday without additional information. **Answer:** Unknown.

1000 D.5.2 FAILED EXAMPLE 2: MACHINE TRANSLATION

1001
1002 For the machine translation task using the CSMT dataset, the model was asked to translate the
1003 Chinese sentence "" into English. The initial translation was as follows:

1004
1005 **Translation:** "He wants to recruit cadres from the same village to go underwater
1006 together and engage in drug trafficking."

1007 The critic model's feedback included one response indicating that the translation was accurate, while
1008 another suggested an alternative translation to better capture the idiomatic meaning. The gold answer
1009 was:

1010
1011 **Gold Answer:** "He wants to take the cadres of the same village to sell drugs with
1012 him."

1013 This example highlights the difficulty in handling idiomatic expressions and the need for a more
1014 nuanced understanding of context during the feedback generation process.
1015

1016 D.5.3 FAILED EXAMPLE 3: PROGRAMMING PROBLEM SOLVING

1017
1018 In the MBPP dataset, the model was tasked with solving a programming problem that required
1019 generating a function to find the volume of a triangular prism. The initial response was:

```
1020 def find_Volume(base, height, length):  
1021     return base * height * length  
1022
```

1023 The critic model provided the following feedback:

1024
1025 **Feedback:** There is no error in the code provided. The function 'find_Volume'
correctly calculates the volume by multiplying the base, height, and length together.

1026 However, the gold answer revealed that the function should calculate the volume of a triangular prism,
1027 which requires dividing the product by 2:
1028

1029 **Gold Answer:** `def find_Volume(l,b,h):`
1030 `return ((l * b * h) / 2)`
1031

1032 The final solution did not incorporate the correct formula for calculating the volume of a triangular
1033 prism, resulting in an incorrect function. This failure illustrates the challenge of correctly interpreting
1034 and implementing mathematical formulas in code generation tasks.

1035 D.6 SUCCESSFUL EXAMPLES AND ANALYSIS

1036 Despite multiple inaccurate feedback samples, a robust FoF process can identify and clarify the errors,
1037 leading to a correct refined feedback and final answer.
1038

1039 In this example, the initial answer is incorrect, and both feedback samples provide wrong suggestions.
1040 However, FoF successfully identifies the errors in both feedback samples, then provides a clear
1041 explanation of the correct formula, demonstrating its ability to identify and clarify errors in the
1042 feedback samples. This leads to a refined feedback that provides the correct answer, showcasing
1043 the robustness of FoF in handling inaccurate feedback and guiding the model towards the correct
1044 solution.
1045

1046 Successful example due to correct FoF clarification

1047 **Gold Answer:**

1048 `def find_rect_num(n):`
1049 `return n * (n + 1)`
1050

1051 **Initial Answer:**

1052 `def find_rect_num(n):`
1053 `return n * (n + 1) * (2 * n + 1) // 6`
1054

1055 **Feedback sample 1:** The answer is using wrong formula, the correct formula should be $(n * (n + 1) // 2) ** 2$. (*Wrong feedback*)
1056

1057 **Feedback sample 2:** The given answer complicates the problem; the answer should be $n ** 2$. (*Wrong feedback*)
1058

1059 Clarifications are needed from the sampling feedback, try to clarify the feedback.

1060 **FoF:** The refined answers provided in both feedback are incorrect. Feedback sample 1 uses
1061 wrong formula. Feedback sample 2 oversimplifies the problem by suggesting $n ** 2$. The correct
1062 formula for the number of rectangles that can be formed using n points on a line is $n * (n + 1)$.
1063 (*FoF correctly identifies the errors in the feedback samples*)

1064 **Refined Feedback:** The correct formula for the question should be $n * (n + 1)$ (*Correct Refined
1065 feedback*)

1066 E PROMPT

1067 F CRITICBENCH EVALUATION METHOD

1068 G FOF ALGORITHM

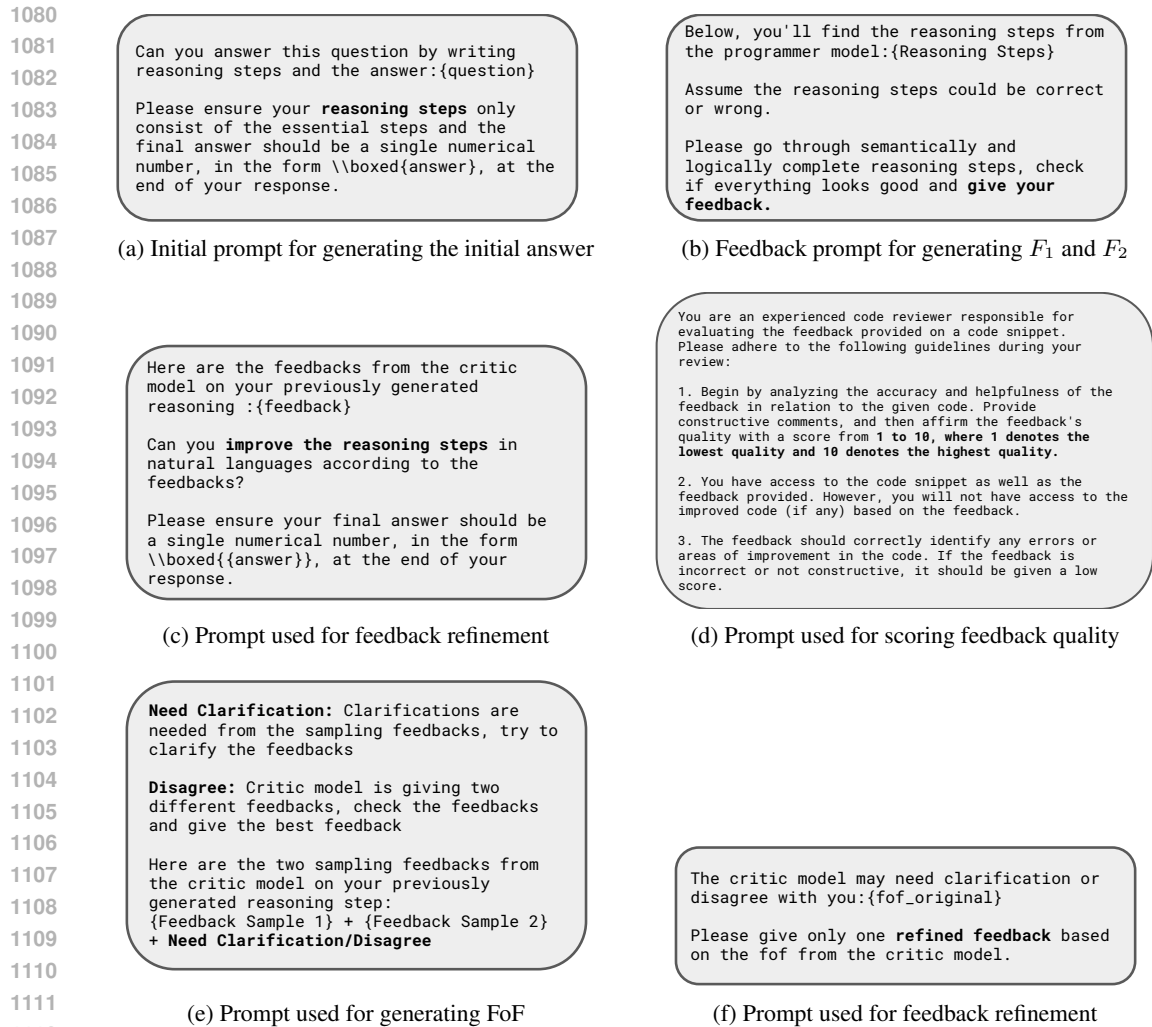


Figure 4: Collection of prompts used for various stages in the feedback generation and refinement process.

1113
1114
1115
1116
1117
1118
1119
1120
1121
1122
1123
1124
1125
1126
1127
1128
1129
1130
1131
1132
1133

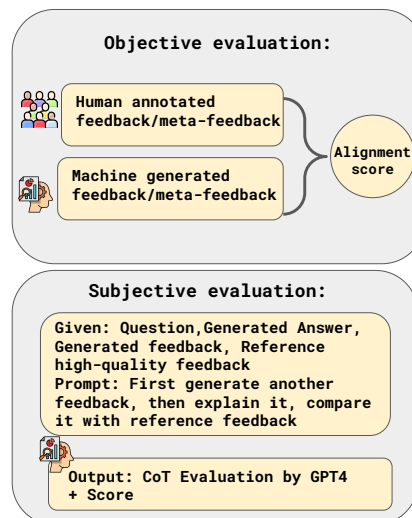


Figure 5: Overview of two evaluation methods in Criticbench

1134
1135
1136
1137
1138
1139
1140
1141
1142
1143
1144
1145
1146
1147
1148
1149
1150
1151
1152
1153
1154
1155
1156
1157
1158
1159
1160
1161
1162
1163
1164
1165
1166
1167
1168
1169
1170
1171
1172
1173
1174
1175
1176
1177
1178
1179
1180
1181
1182
1183
1184
1185
1186
1187

Algorithm 1 FoF Algorithm

Require: Question Q , Base Generator BG , Critic Model CM , Semantic Similarity Thresholds θ_1, θ_2 , Feedback Rounds
Ensure: Final Answer R_f

- 1: $R_0 \leftarrow BG(p_{gen} \parallel Q)$ ▷ Initial generation (Eqn. ??)
- 2: **while** Round < Feedback Rounds **do** ▷ Iterative refinement loop
- 3: $F_1, F_2 \leftarrow CM(p_{fb} \parallel Q, R_0)$ ▷ Feedback generation (Eqn. ??)
- 4: $S \leftarrow \text{SemanticSimilarity}(F_1, F_2)$
- 5: **if** $S < \theta_1$ or $\theta_1 < S < \theta_2$ **then** ▷ If feedback 1 and 2 disagree with each other or clarification needed
- 6: $FoF \leftarrow BG(p_{fof} \parallel F_1, F_2)$ ▷ FoF generation (Eqn. ??)
- 7: $RF \leftarrow CM(p_{rf} \parallel Q, R_0, FoF, F_1, F_2)$ ▷ Refine feedback (Eqn. ??)
- 8: **else**
- 9: $RF \leftarrow F_1$ ▷ Use first feedback
- 10: **end if**
- 11: $R_f \leftarrow BG(p_{fof} \parallel Q, R_0, RF)$ ▷ Refine initial answer (Eqn. ??)
- 12: **if** RF contains "this answer is correct" **then** ▷ Check for stop condition
- 13: **return** R_f
- 14: **end if**
- 15: $R_0 \leftarrow R_f$ ▷ Update initial answer for the next iteration
- 16: Round \leftarrow Round + 1 ▷ Increment round counter
- 17: **end while**
- 18: **return** R_f ▷ Return final answer after maximum rounds
