Project Part 1

Generative AI

Saarland University – Winter Semester 2024/25

Gaygusuz Osman
7065438
`osga00001@stud.uni-saarland.de`

1 Exercise Implementation: I1

The results obtained for generating repairs using the `gpt-4o-mini` are:

Problem	RPass (%)	REdit
Problem 1	80.0	15.50
Problem 2	80.0	33.50
Problem 3	100.0	16.60
Problem 4	80.0	23.25
Problem 5	60.0	18.33

Table 1: Repair Generation Results Per Problem Using `gpt-4o-mini`

Metric	Value
Overall Correct Rate (%)	80.0
Overall Avg. Edit Distance	21.35

Table 2: Overall Repair Generation Results Using `gpt-4o-mini`

2 Exercise Implementation: I2

The results for hint generation using the `gpt-4o-mini` model obtained are below:

Problem	HCorrect (%)	HInformative (%)	HConceal (%)	HComprehensible (%)	HGood (%)
Problem 1	100.0	100.0	40.0	100.0	40.0
Problem 2	40.0	60.0	40.0	100.0	40.0
Problem 3	100.0	80.0	100.0	100.0	80.0
Problem 4	100.0	100.0	100.0	80.0	80.0
Problem 5	80.0	100.0	80.0	100.0	80.0

Table 3: Hint Generation Results Per Problem Using `gpt-4o-mini`

One of the main problem of the model is to not give the answer in the hint and try to adjust it to let the learner think to fix.

Metric	Value (%)
HCorrect	84.0
HInformative	88.0
HConceal	72.0
HComprehensible	96.0
HGood	64.0

Table 4: Overall Hint Generation Results Using `gpt-4o-mini`

3 Exercise Implementation: I3

The results obtained for generating repairs using the `Phi-3-mini` model bellow:

Problem	RPass (%)	REdit
Problem 1	40.0	14.00
Problem 2	00.0	-1
Problem 3	20.0	10.00
Problem 4	80.0	22.50
Problem 5	40.0	17.50

Table 5: Repair Generation Results Per Problem Using `Phi-3-mini`

Metric	Value
RPass (%)	36.0
REdit	18.1

Table 6: Overall Repair Generation Results Using `Phi-3-mini`

4 Exercise Implementation: I4

The results for hint generation using the `phi-3-mini` model are presented below.

Problem	HCorrect (%)	HInformative (%)	HConceal (%)	HComprehensible (%)	HGood (%)
Problem 1	60.0	60.0	100.0	100.0	60.0
Problem 2	40.0	60.0	100.0	100.0	40.0
Problem 3	100.0	80.0	100.0	100.0	80.0
Problem 4	80.0	80.0	100.0	80.0	80.0
Problem 5	60.0	80.0	100.0	80.0	60.0

Table 7: Hint Generation Results Per Problem Using `phi-3-mini`

Metric	**Value (%)**
HCorrect	68.0
HInformative	72.0
HConceal	100.0
HComprehensible	92.0
HGood	64.0

Table 8: Overall Hint Generation Results Using `Phi-3-mini`

5 Exercise Implementation: I5

Problem	RPass (%)	REdit
Problem 1	80.0	12.75
Problem 2	100.0	33.00
Problem 3	100.0	16.60
Problem 4	100.0	20.80
Problem 5	100.0	14.25

Table 9: Repair Generation Results Per Problem Using `gpt-4o-mini`

Metric	Value
Overall Correct Rate (%)	92.0
Overall Avg. Edit Distance	22.34

Table 10: Overall Repair Generation Results Using `gpt-4o-mini`

6 Exercise Implementation: I6

Problem	RPass (%)	REdit
Problem 1	80.0	14.50
Problem 2	20.0	01.00
Problem 3	40.0	29.00
Problem 4	80.0	23.00
Problem 5	40.0	17.00

Table 11: Repair Generation Results Per Problem Using `phi-3-mini`

Metric	Value
Overall Correct Rate (%)	52.0
Overall Avg. Edit Distance	18.69

Table 12: Overall Repair Generation Results Using `phi-3-mini`

7 Exercise Implementation: I7

The results for hint generation using the `gpt-4o-mini` model are presented below.

Problem	HCorrect (%)	HInformative (%)	HConceal (%)	HComprehensible (%)	HGood (%)
Problem 1	80.0	80.0	100.0	100.0	80.0
Problem 2	80.0	80.0	100.0	100.0	80.0
Problem 3	100.0	100.0	100.0	100.0	100.0
Problem 4	100.0	100.0	100.0	100.0	100.0
Problem 5	100.0	100.0	100.0	100.0	100.0

Table 13: Hint Generation Results Per Problem Using `gpt-4o-mini`

Metric	Value (%)
HCorrect	92.0
HInformative	92.0
HConceal	100.0
HComprehensible	100.0
HGood	92.0

Table 14: Overall Hint Generation Results Using `gpt-4o-mini`

8 Exercise Implementation: I8

The results for hint generation using the `gpt-4o-mini` model are presented below. The table summarizes the average metrics for each problem and overall.

Table 15: Hint Generation Results Per Problem Using `phi-3-mini`

Problem	HCorrect (%)	HInformative (%)	HConceal (%)	HComprehensible (%)	HGood (%)
Problem 1	80.0	80.0	100.0	100.0	80.0
Problem 2	60.0	80.0	100.0	100.0	60.0
Problem 3	80.0	80.0	100.0	100.0	80.0
Problem 4	100.0	100.0	100.0	100.0	100.0
Problem 5	80.0	80.0	100.0	100.0	80.0

One of the main problem of the model is to not give the answer in the hint and try to adjust it to let the learner think to fix.

Table 16: Overall Hint Generation Results Using `phi-3-mini`

Metric	Value (%)
HCorrect	80.0
HInformative	84.0
HConceal	100.0
HComprehensible	100.0
HGood	80.0

9 Exercise Implementation: I9

Since the hint depends on the accuracy of the repaired program, we must ensure that the repaired program is as close to correct as possible. Otherwise, it would generate false hints for issues that are irrelevant. Another way to improve the results is to split the prompts into separate tasks, preventing the model from being overwhelmed by trying to do too many things at once.

Initially, the approach did not involve explicitly using the repaired program to guide hint generation. This led to generic hints that were not tailored to the specific corrections in the repaired code. Moreover, without the repaired program, the model had to rethink the corrections from scratch, which added unnecessary complexity. To address this, I tried adding the best-repaired program to the prompt for GPT-4o-mini, which resulted in significant improvements.

Additionally, the workflow lacked a proper connection between the explanation (chain-of-thought) and the hint generation. If we ask the model to generate both a chain-of-thought explanation and a hint in a single prompt, we cannot leverage the explanation effectively for hint generation. This is because the model's knowledge and context remain the same for both tasks. Instead, splitting the tasks into two steps—first generating the explanation and then using it to generate the hint in a second prompt—improves results. However, this approach has a limitation: calling the model multiple times increases computational costs.

I also observed that modifying the prompt for repair generation was essential, as the initial repair prompt was not detailed enough, which heavily impacted results. Previously, the prompt simply asked to "fix the buggy code." This could lead the model to create entirely new solutions, sometimes solving the problem in a different way than intended. While the solution might work, it could generate a hint that is out of context or overly complex, requiring significant effort from the student to adapt their code. To address this, I modified the prompt to request minimal changes to fix the buggy program. This ensures the repaired program remains as close as possible to the original, making it easier for the student to understand both the solution and the underlying problem.

After obtaining the repaired program, I introduced another prompt to generate a chain-of-thought explanation describing how and why the repaired program resolves the issue. To achieve this, I created a method to extract the explanation from the first response. Finally, I called the model again to generate the hint using the chain-of-thought explanation.

Unfortunately, while doing this, I encountered a new issue: a Value Error indicating that the input length (3101) combined with max new tokens (2048) exceeded the maximum sequence length (4096). The problem arose because I included the problem description, buggy code, repaired code, and explanation in the second prompt. To resolve this, I removed the repaired code from the second prompt, relying solely on the explanation since it was generated from the repaired code. However, this was still insufficient, as the input length was reduced to 2895, and the issue persisted. The main cause was that the explanation itself was too long. Thus, i asked in the prompt to generate the explanation COT in a few sentences to not generate a long paragraph. All of this improved the results significantly.

Acknowledgements

Used for the implementation I7

Used for the implementation I7