Comparative Evaluation of RAG-Based and Non-RAG Methods for Teaching OOP Topic

Overview of Evaluation Dimensions

To compare the RAG-based and non-RAG methods for generating problem statements, evaluation rubrics, and feedback, I analyzed the outputs across the following criteria:

1. Problem Statement Quality

- Clarity: How clearly is the problem defined?

- Comprehensiveness: Does the problem statement cover the intended topic thoroughly?

- Applicability: Is the problem relevant to real-world scenarios or professional contexts?

2. Evaluation Rubric

- Specificity: Are the criteria detailed enough to ensure consistent assessment?

- Relevance: Do the criteria align with the learning goals of the topic?

3. Feedback Quality

- Constructiveness: Does the feedback offer actionable insights for improvement?

- Depth: Does it identify both strengths and weaknesses in the student's solution?

- Encouragement: Does it foster a positive learning experience?

Topic 0: Inheritance and Polymorphism in C++

RAG-Based Method:

- Problem Statement: Well-defined and practical, requiring a class hierarchy to calculate geometric shapes' areas. The focus on polymorphism and inheritance was explicit, with a real-world application (graphics).

- Evaluation Rubric: Highly specific and detailed, assessing inheritance, polymorphism, and abstraction.

- Feedback: Constructive but slightly generic, encouraging deeper exploration of OOP principles like encapsulation.

Non-RAG Method:

- Problem Statement: Clear but less challenging, focusing on a simple class hierarchy with a single polymorphic behavior.
- Evaluation Rubric: Adequate but less detailed, primarily focusing on correct implementation and readability.
- Feedback: Detailed and appreciative of the solution but lacked depth in encouraging improvements.

Verdict: The RAG-based method outperformed in crafting a comprehensive and challenging problem with a better rubric.

Topic 1: Encapsulation and Abstraction

RAG-Based Method:

- Problem Statement: A robust design task involving shape positioning and color encapsulation, requiring abstraction and inheritance.
- Evaluation Rubric: Comprehensive, addressing encapsulation, abstraction, and maintainability through setters and getters.
- Feedback: Focused on encouraging better encapsulation and abstraction, with specific suggestions.

Non-RAG Method:

- Problem Statement: Simpler, involving a basic BankAccount class to demonstrate encapsulation.
- Evaluation Rubric: Minimalist, focusing primarily on private data and public methods.

- Feedback: Appreciative of encapsulation but didn't highlight areas for extending abstraction.

Verdict: The RAG-based method offered a more engaging and realistic problem, with better feedback on improving the solution.