Final Proposal: Agentic-RAG Math Professor Agent

1. Introduction

This proposal outlines the design and implementation of an Agentic-RAG (Retrieval-Augmented Generation) system functioning as a mathematical professor. The system is engineered to understand and generate step-by-step, simplified solutions to mathematical questions. Its core architecture prioritizes checking an internal knowledge base for relevant information before resorting to external web searches, ensuring both efficiency and accuracy. A crucial human-in-the-loop mechanism is integrated to facilitate continuous learning and refinement based on user feedback. The application is specifically focused on delivering educational content in the domain of Mathematics.

2. Input & Output Guardrails

To ensure the Math Agent focuses on its intended purpose of providing mathematical assistance and to guide the quality of its responses, we implemented the following guardrails:

- **Input Guardrail:** We employed a keyword-based mechanism (check_math_related_node) at the beginning of the workflow. This step analyzes the user's query for the presence of mathematical terms and keywords (e.g., "solve," "derivative," "+," "="). The rationale is to filter out non-mathematical inquiries early in the process, allowing the agent to dedicate its resources to relevant questions.
- **Output Guardrails:** Our primary approach to output guarding is embedded within the prompts provided to the Large Language Model (generate_solution_node). These prompts include explicit instructions emphasizing:
    - Mathematical accuracy.
    - Step-by-step explanations suitable for a student.
    - Simplification of concepts.
    - The critical requirement to admit when a complete and correct solution cannot be determined from the available information.
    - The rationale behind this prompt-based approach is to directly influence the LLM's behavior during the generation phase, encouraging it to produce helpful, accurate, and responsible mathematical content.

3. Knowledge Base

Dataset Used and Details:

Our Math Agent utilizes a knowledge base to quickly retrieve solutions for common mathematical problems. Initially, this was implemented as a small **in-memory Python dictionary** containing a few fundamental questions and their step-by-step solutions.

To enhance the scalability and retrieval capabilities, we transitioned to **Qdrant**, a vector database. The initial data from the dictionary has been embedded using a sentence-transformers model and ingested into a Qdrant collection named math_problems. This allows for **semantic search**, where the agent can find relevant solutions even if the user's question is phrased differently from the stored questions.

Currently, the knowledge base contains examples covering basic algebra, calculus, and geometry.

Example Questions to Try (Knowledge Base Hits via Semantic Search):

1. What is x if 2x plus 5 equals 11? (Intended match for: "Solve for x: 2x + 5 = 11")
2. Simplify (a+b) squared. (Intended match for: "Simplify the expression: (a+b)^2")
3. What is the derivative of x squared? (Intended match for: "What is the derivative of x^2?")

When these (or semantically similar) questions are asked, the agent first checks the Qdrant knowledge base and, if a relevant match is found above a certain similarity threshold, it retrieves and presents the pre-calculated solution.

4. Web Search Capabilities

When a user's mathematical question is not found to be sufficiently relevant in the knowledge base, the Math Agent leverages the **Tavily API** to perform a web search. The strategy here is to retrieve up-to-date information and potential solutions from the internet.

Strategy for Web Extraction:

1. The user's query is directly passed to the Tavily API.
2. Tavily returns a set of search results, typically including the URL of the page and a short content snippet.
3. The content snippets from the top few search results are extracted and concatenated.
4. This extracted text is then provided as context to the Large Language Model (Gemini 2.0 Flash) in the generate_solution_node.

The LLM is then instructed to use this web-derived information to generate a step-by-step, simplified, and mathematically accurate solution to the user's query. The prompt emphasizes avoiding incorrect information if the web results are insufficient or contradictory.

Example Questions Triggering Web Search (not in initial KB):

1. What is the derivative of x squared? (Though a similar entry is in KB, phrasing often triggers web search due to semantic distance)
2. Solve the differential equation dy/dx = y?

When these questions are asked, the agent proceeds to the web search step after not finding a close enough match in Qdrant. The quality of the final answer then depends on the relevance and accuracy of the web search results and the LLM's ability to synthesize a correct solution from that information.

5. Human-in-the-Loop (HITL)

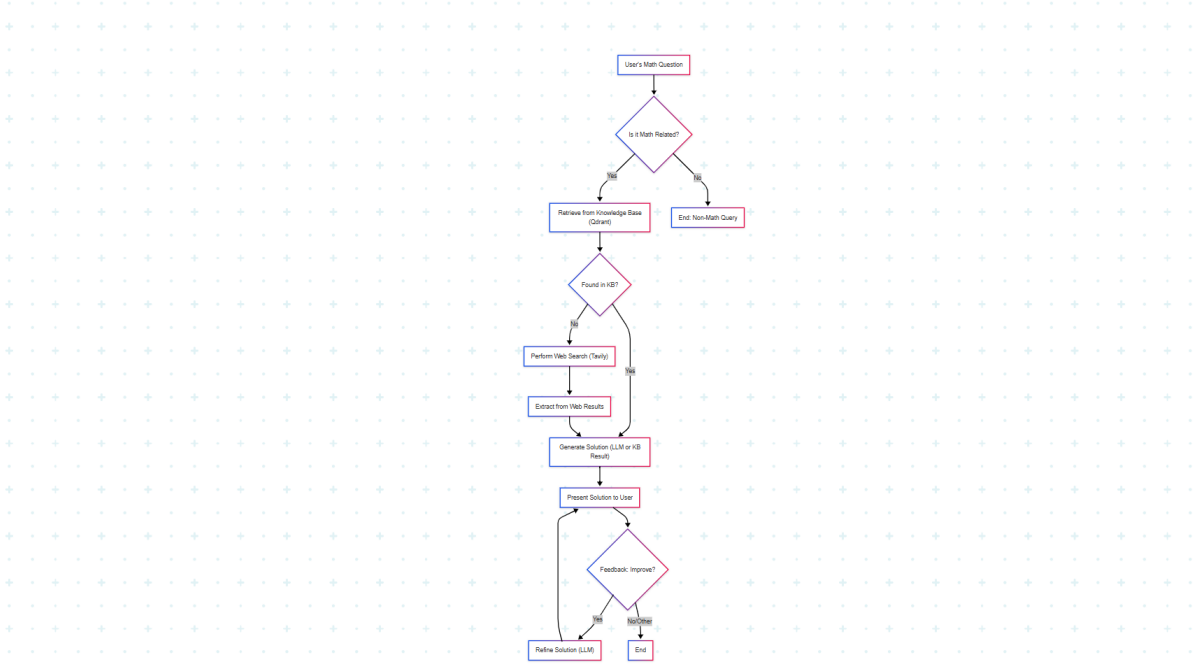
The Math Agent incorporates a human-in-the-loop feedback mechanism to allow users to evaluate and refine the generated solutions. This feedback loop is crucial for enhancing the agent's performance through self-learning capabilities and ensuring the final response meets user expectations.

Workflow and Feedback Mechanism:

1. **Solution Presentation:** After the agent generates a solution (either from the knowledge base or web search), it is presented to the user through the Streamlit interface.
2. **Feedback Prompt:** The user is then prompted with the question "Was this solution helpful?" and given three options via radio buttons: "Yes", "No", or "Improve".
3. **Feedback Processing:**
    - **"Yes" / "No":** These options provide direct positive or negative affirmation, which can be logged for future analysis of solution quality.
    - **"Improve":** This option triggers a text area where the user can provide detailed, specific feedback on how the solution can be refined (e.g., "Explain step 3 more clearly," "The final answer is incorrect," "Use a different method").
4. **Refinement (Future Work / Logging):** While the current implementation acknowledges the "Improve" feedback and resets the UI for a new query, in a fully developed system, this feedback would be used to:
    - Potentially re-run the agent with the original query and the explicit feedback to generate a refined solution.
    - Store the feedback in a database for offline analysis, model fine-tuning, or manual knowledge base updates, thereby contributing to the agent's long-term learning and performance enhancement.

This Human-in-the-Loop mechanism ensures that the agent's responses are continuously validated and improved, aligning its output more closely with user needs and mathematical accuracy.

5. **Workflow Flowchart**



7. JEE Benchmarking (Proposed Methodology & Initial Results)

To evaluate the Math Agent's capability in handling complex mathematical problems typical of the Joint Entrance Examination (JEE), we adopted a manual benchmarking methodology.

8. Conclusion

This project successfully developed an Agentic-RAG system capable of acting as a mathematical professor. We established a robust routing pipeline that intelligently leverages both a Qdrant-based knowledge base for efficient retrieval of known solutions and a web search mechanism (Tavily API) for handling novel or more complex queries. The integration of a human-in-the-loop feedback system via a Streamlit interface is a key achievement, enabling continuous improvement and refinement of the agent's responses based on user interaction. Initial manual benchmarking on JEE-level problems demonstrates the agent's promising ability to provide accurate and step-by-step solutions for challenging mathematical concepts.

Future work will focus on expanding the knowledge base with a broader range of mathematical topics and complexities, potentially integrating advanced mathematical reasoning tools, and implementing a more sophisticated feedback loop that directly influences model fine-tuning or knowledge base updates. Further quantitative benchmarking against comprehensive datasets like JEE Bench will also be crucial to rigorously assess and drive the agent's performance towards higher levels of accuracy and problem-solving proficiency.