CourseGPT-Pro

Project Milestone 1: Problem Definition & Literature Review

Project Title: *CourseGPT-Pro: A Multimodal, Tool-Augmented RAG System for Technical Homework Assistance*

Abstract

Students working on technical assignments in engineering and computer science face a major challenge: course content is multimodal, featuring a complex combination of text, mathematical expressions (LaTeX), diagrams, tables, and code segments. Current AI helpers and naive Retrieval-Augmented Generation (RAG) models fall short, as they cannot interpret this structural complexity or perform the required logical and computational reasoning. This project proposes **CourseGPT-Pro**, an end-to-end system designed to overcome these limitations. By integrating a high-fidelity multimodal ingestion pipeline, a **hybrid retrieval system**, a **tool-augmented generative model orchestrated by a router and sub-agents**, and a final **verification layer**, CourseGPT-Pro aims to deliver accurate, reliable, and fully-cited answers to challenging technical problems, directly filling the gaps left by existing solutions.

1. Project Objectives

The main goal is to build, train, and test a reliable homework bot that delivers verifiable, step-by-step assistance on technical university problems (e.g., IIT Madras curriculum).

To accomplish this, we have outlined the following specific objectives:

- **Objective 1:** High-Fidelity Multimodal Ingestion
  To create a deep learning pipeline that parses PDF documents (lecture slides, textbooks) into a structured format. This includes extracting text, correctly transcribing LaTeX equations, recovering table structures, identifying code blocks, and linking figures to their captions.

- **Objective 2:** Deterministic OCR for Grounding & Safety
  To establish a canonical OCR text layer (tokens + bounding boxes + reading order) as the foundation for document understanding.
  - **Rationale:** While multimodal models like LayoutLMv3 are excellent for understanding layout and semantics, a deterministic OCR layer provides **symbol-level ground truth**. This is critical for:
    1. **Safety & Reliability:** Agents (math, code) operate on exact, verifiable text and LaTeX, not paraphrased interpretations, reducing the risk of executing on misread inputs.
    2. **Verifiability:** It enables precise **page-and-region citations**, allowing users to trace every piece of information to its exact source.
    3. **Hybrid Search:** It powers exact lexical search for specific terms, variable names, and theorem IDs, which complements dense vector search.
- **Objective 3:** Context-Aware Hybrid Retrieval
  To develop a retrieval system that blends multiple search modalities for superior context. This "hybridness" involves combining:
  - Dense vector search for conceptual understanding.
  - Sparse lexical search (BM25) for keywords and symbols from the OCR layer.
  - Specialized search for math (LaTeX-aware) and code (AST-aware).
  - Vision search for diagrams and figures based on image embeddings and captions.
- **Objective 4:** Strong Tool-Augmented Reasoning
  To train a generative language model to reliably call external tools. The model must learn when to delegate tasks to a symbolic math solver (e.g., SymPy) for exact computation and a sandboxed code interpreter for running program logic, then integrate the results seamlessly into its explanation.

- **Objective 5:** Verify & Trust
  To build a verification module that cross-checks the generated answer against retrieved sources. The output must feature exact, page-and-region-level citations to ensure user trust and traceability.

- **Objective 6:** Comprehensive Performance Assessment
  To quantify component and end-to-end system performance using a combination of automated metrics and evaluation by powerful "judge" models on a vetted set of homework problems.

2. Literature Review & Existing Solutions

Our work extends research in four main areas: Retrieval-Augmented Generation (RAG), Multimodal Document Understanding, Tool-Augmented Language Models, and Factuality in AI.

A. **Retrieval-Augmented Generation (RAG):** The paradigm of marrying pre-trained LMs with external retrieval. Libraries like LangChain and LlamaIndex provide frameworks for basic RAG.

B. **Multimodal Document Understanding:** Models like **LayoutLMv3** and **Donut** can understand document geometry without external OCR. We use these to *augment* our foundational OCR layer, using them to identify logical regions (e.g., paragraphs, captions) that contain the OCR-extracted text.

C. **Tool-Augmented Language Models:** Frameworks like **Toolformer** and **ReAct** provide robust patterns for interleaving reasoning with tool calls. Modern models have native function-calling capabilities.

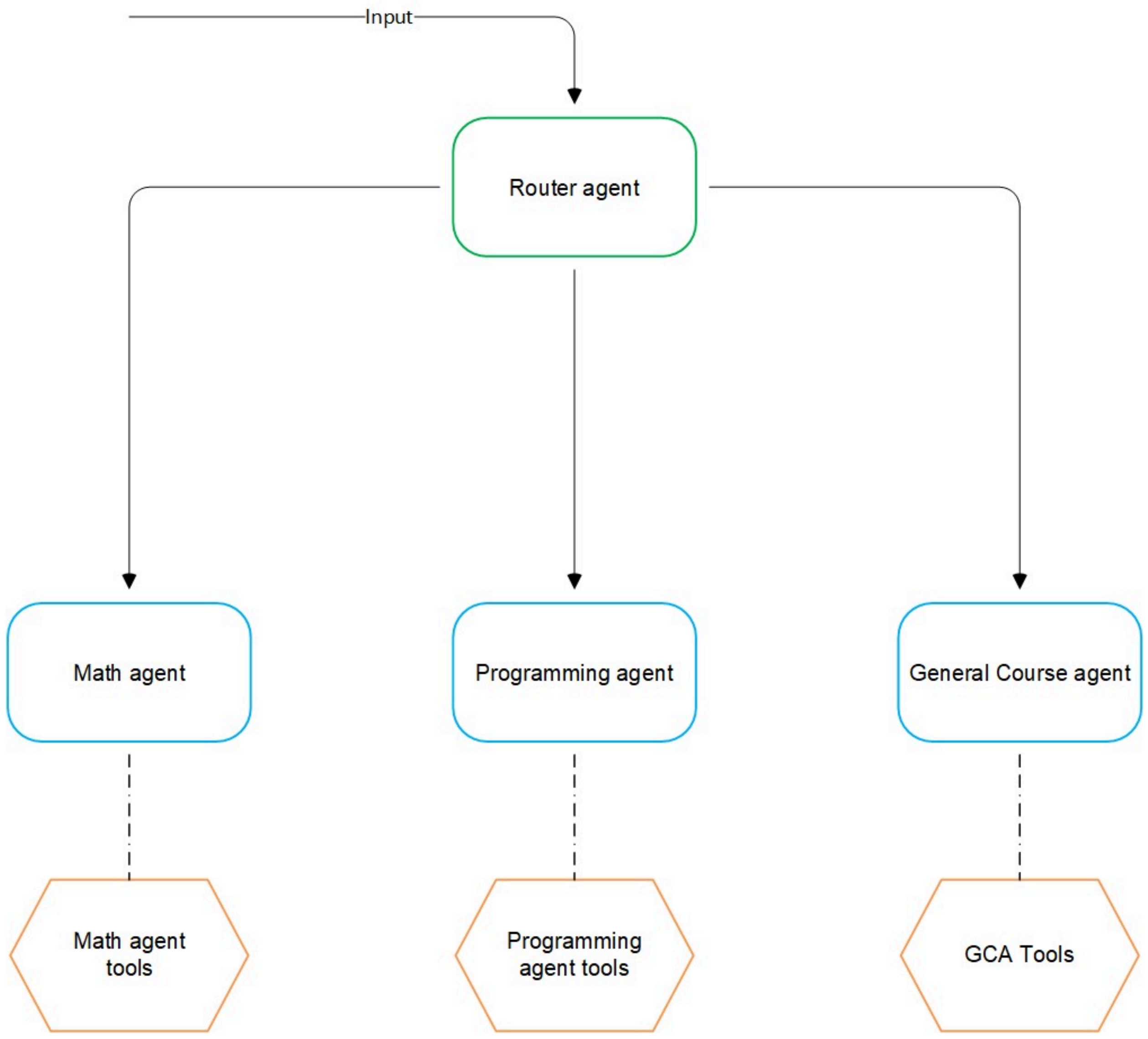
D. **Verifiability and Faithfulness:** Techniques like citation generation and Natural Language Inference (NLI) checks are used to reduce hallucination and ground claims in source data.

3. System Overview & Architecture

3.1. Agentic Orchestration: Router + Sub-Agents

The system is orchestrated by an intelligent, multi-agent architecture. This approach avoids using a single, monolithic model, instead routing tasks to specialized agents for higher accuracy and efficiency.

- **Router Agent:** A fast and efficient model (e.g., **Llama 3.2**, **DeepSeek-V2-Lite**, **GLM-4-9B-Chat**) acts as the central coordinator. It receives the user's query, analyzes it, and creates a plan. It then delegates sub-tasks to the appropriate agent.
- **Sub-Agents:**
  - **Math Agent:** Responsible for solving mathematical problems. It uses planning and reasoning to invoke a **symbolic math engine (SymPy)** for precise calculations (algebra, calculus, etc.).
  - **Code Agent:** Executes code snippets in a secure, sandboxed environment to verify logic, run tests, or generate outputs.
  - **RAG & Search Agent:** Performs hybrid retrieval over the ingested document knowledge base to find relevant definitions, theorems, and contextual paragraphs. It can also perform web searches if external knowledge is required.
- **Workflow:** The Router receives a query (e.g., "Solve problem 3.5 from the textbook"). It first uses the RAG agent to retrieve the problem text and any relevant theory. It then passes the mathematical part to the Math Agent. If the problem involves code, the Code Agent is also invoked. The Router collects the outputs from all sub-agents and synthesizes them into a final, coherent, step-by-step solution.



4. Datasets & Licensing

We will leverage existing, high-quality datasets for training and evaluation. All selected datasets have permissive, **OSI-approved licenses (e.g., MIT, Apache 2.0)** or allow for non-commercial research use. We do not plan to create a new dataset from scratch but may use knowledge distillation from leading models to generate synthetic training data if required.

- **MathVista:** CC BY-SA 4.0 (Evaluation, Non-commercial)
  https://huggingface.co/datasets/AI4Math/MathVista
- **DAPO-Math-17k:** Apache-2.0 (Permissive, OSI-approved)

https://huggingface.co/datasets/BytedTsinghua-SIA/DAPO-Math-17k

- **MathX-5M:** MIT (Permissive, OSI-approved)

  https://huggingface.co/datasets/XenArcAI/MathX-5M

- **ScienceQA:** CC BY-SA 4.0 (Evaluation, Non-commercial)

  https://huggingface.co/datasets/armanc/ScienceQA

- **JEE-NEET Benchmark:** MIT (Permissive, OSI-approved)

  https://huggingface.co/datasets/Reja1/jee-neet-benchmark

- **JEEBench:** MIT (Permissive, OSI-approved)

  https://huggingface.co/datasets/daman1209arora/jeebench

5. Baselines & Performance Assessment

5.1. Baselines

To measure the effectiveness of our system, we will compare it against several baselines:

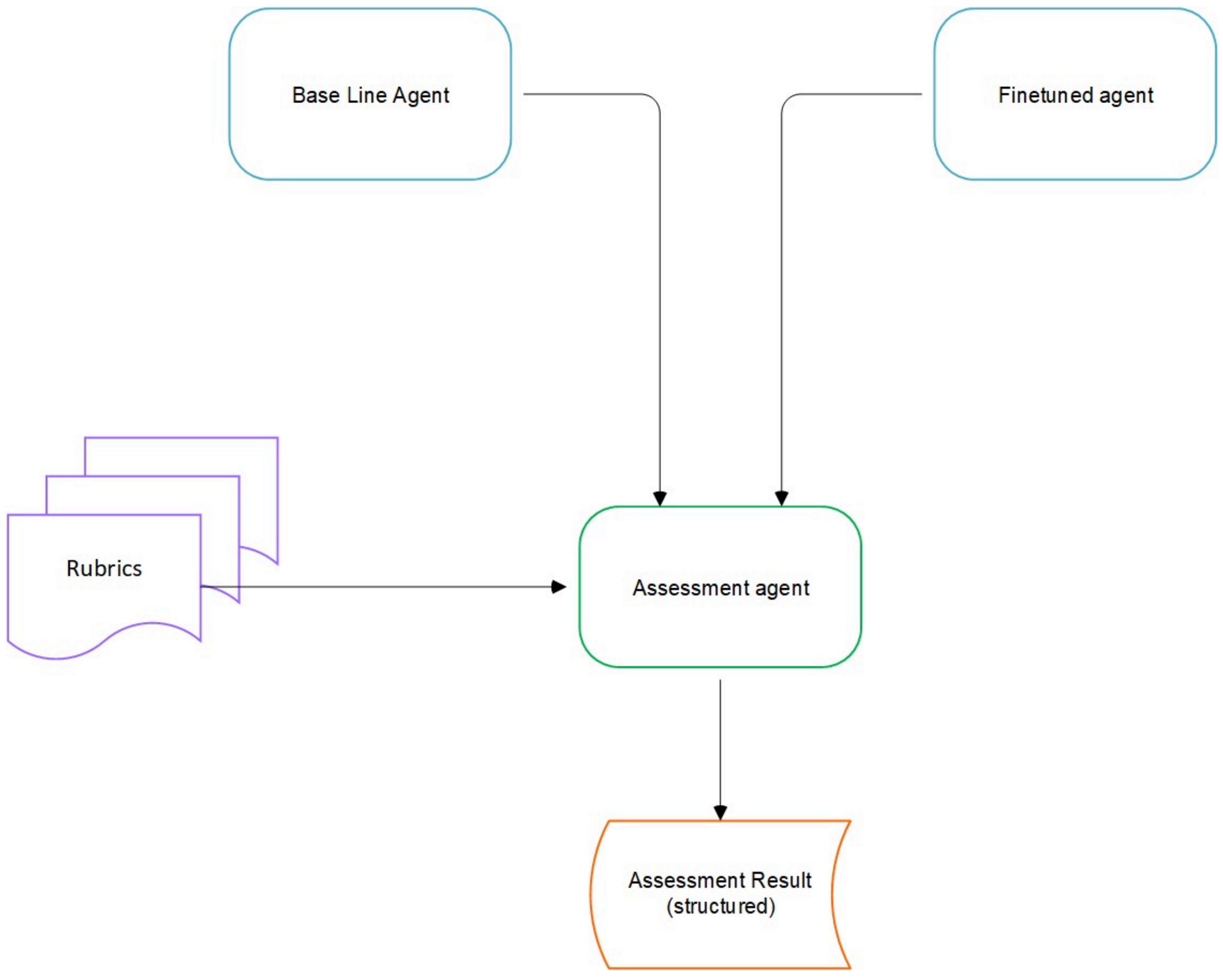
- **Baseline 0 (Naïve RAG):** A simple text-only pipeline using a sentence-splitter, vector DB, and a generic instruction-tuned model.
- **Baseline 1 (Pragmatic SOTA):** An end-to-end system using a fast and capable proprietary model like **Gemini 2.5 Flash-Lite**. This provides a reference for speed, cost, and out-of-the-box performance.

5.2. Model Performance Assessment

We will employ a robust "judge-as-a-benchmark" methodology for evaluation.

- **Judge Models:** For questions where ground truth is complex (e.g., multi-step proofs), we will use powerful, frontier models (e.g., **Gemini 2.5 Pro**, **GPT-5**, **Grok-4**) as expert evaluators.

- **Rubric-Based Evaluation:** The judge model will be given the original question, the ground truth answer (if available), and the generated answer from our system. It will be prompted to score the output on a predefined rubric (e.g., correctness, clarity, citation accuracy, step-by-step logic) and provide its reasoning in a **structured output format (JSON)**.
- **Metrics:** This process will yield quantitative benchmarks for accuracy, faithfulness, and overall quality, allowing for rigorous comparison against baselines.



6. Development Platform

For the initial phases of development and experimentation, we plan to utilize accessible, cloud-based notebook environments.

- **Primary Platforms:** Development will primarily be conducted on **Google Colab** and **Kaggle Notebooks**. These platforms provide free access to powerful GPUs (e.g., NVIDIA T4, P100), which is sufficient for fine-tuning models, running experiments, and developing the core components of our system without incurring initial infrastructure costs.
- **Contingency for Scale:** If the project requires more substantial or sustained computational resources (e.g., for larger model training or hosting persistent endpoints for demonstration), we will leverage the **Google Cloud Platform (GCP)**. Specifically, we may use **Vertex AI** for managed training jobs and model deployment, funded by available free trial credits.