Details

- Full Name : Ravindu Hiran Weerakoon
- Email: ravinduhiran26@gmail.com
- Github Username : RavinduWeerakoon
- First Language: Sinhala
- Time Zone: Colombo, Sri Lanka GMT+5:30

As a graduate candidate in Computer Science and Engineering from the University of Moratuwa, I possess a robust skill set encompassing Python-based web development, automation, containerization with Docker and Kubernetes, and developing Large Language Model (LLM) applications.

Previous Projects that I have worked with

CVAT-ai

- Fixing a CVAT SDK issue when uploading the resources of path types
    - Languages: Python, Javascript
    - This was a g fix for a file upload issue in CVAT sdk
    - PR: https://github.com/cvat-ai/cvat/pull/9114

- Fixing an Extra Slider issue in a couple of pages
    - Languages: Javascript
    - This involved removing some overflow issues in the CVAT UI
    - PR:https://github.com/cvat-ai/cvat/pull/9168

Ballerina/WSO2

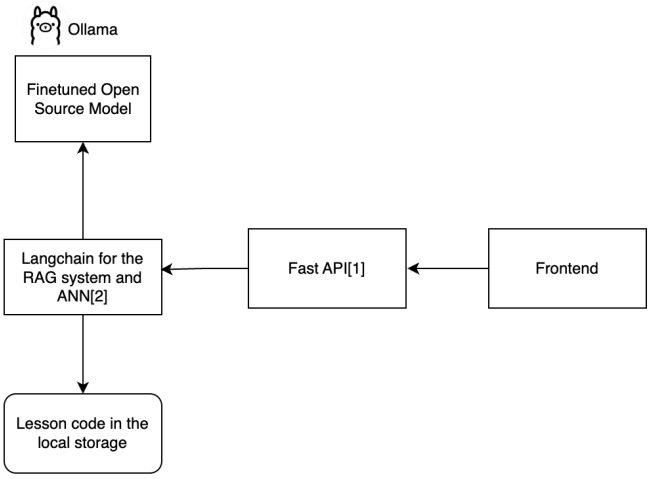
- Creating a ballerina module to connect with Hubspot CRM Deals
    - Languages: Ballerina, Java
    - PR: https://github.com/ballerina-platform/module-ballerinax-hubspot.crm.object.deals

DataLoom/C2SI

- Adding Docker support for the Dataloom Project and fixing a bug in the Data Upload Page
    - Languages: Python, Docker, Javascript
    - PR: https://github.com/c2siorg/dataloom/pull/30

AI Code generation for lesson plans and model abstraction layer

The core of your project revolves around enabling users to generate Music Blocks code through natural language interactions. Here's a step-by-step breakdown of how it's intended to create AI Code generation for lesson plans in music blocks



Main architecture

Key Components and Their Roles

- **LangChain Based RAG system[2]:**
  - **Managing the FAISS Vector Store and ANN:** Efficiently retrieves relevant data based on semantic similarity, improving the accuracy and relevance of the generated code.
  - The conversational chains in LangChain allow the AI to maintain context and engage in multi-turn conversations with the user. Langchain Provides a consistent interface for interacting with different LLMs, making the system adaptable and future-proof.
- **FastAPI Backend[1]:**
  - API for communication between the UI and the LLM.
  - Processes and routes data between the UI, LangChain, and the LLM.
  - FastAPI is designed for high performance and scalability, ensuring the system can handle multiple users.
- **UI (JavaScript):**
  - Provides an intuitive and user-friendly interface for interacting with the AI.
  - Facilitates natural language conversations with the AI.
  - Displays the generated Music Blocks code in a clear and understandable format.

Model Agnosticism (LangChain and Ollama's' Role)

- Both the ollama and LangChain combination give us a model abstraction layer which is crucial for achieving model agnosticism.
- It provides a standardized interface for interacting with different LLMs, regardless of their underlying architecture.
- So we can easily switch between different LLMs (e.g., Llama, QWEn, Mistral) without making significant changes to the rest of your codebase.

How will it impact Sugar Labs?

This project will significantly enhance the Music Blocks platform by:

- **Improving Lesson Plan Generation:** Enabling the integration of AI-generated code snippets into lesson plans, providing clearer and more interactive learning experiences.
- **Increasing Accessibility:** Making Music Blocks more accessible to users with varying levels of coding experience by simplifying the process of creating projects.
- **Promoting Innovation:** Fostering innovation by providing a flexible and extensible AI framework that can be adapted to future advancements in AI technology.
- **Strengthening the Community:** Providing well-documented code, and guides, that will make it easier for new contributors to join the project.
- The software will be more easy to use as the LLm will help understand the software and will be able to make amazing things out of it

Technologies that I would be using

- Python(Langchain, FastAPI): to create the API endpoints and the model layer abstractions
- JavaScript: To handle the frontend
- Ollama: To run the models(this was the easiest way)
- Unisloth: To finetune the models based on a custom dataset to increase the reasoning ability about the music block environment
- Small parameter model: like QWEn 2.5 3B or llama Coder

My plan

Week 1: Project Setup and Data Preparation

- Set up the development environment, including Python, FastAPI, Ollama, and Unsloth.
- Begin expanding the dataset by gathering additional Music Blocks lesson plans and project data.
- Familiarize myself with the Music Blocks codebase and the environment.
- Start the fine-tuning process after creating the dataset

References that I would be using

https://github.com/meta-llama/llama-cookbook/tree/main

https://huggingface.co/datasets/yahma/alpaca-cleaned

Week 2-3: Model Selection and Initial Training

- Select and configure the chosen small parameter LLM (e.g., QWEn 2.5 Coder 3B or Codellama) using Ollama.
- Perform initial fine-tuning of the LLM using the expanded dataset with Unsloth.
- In this stage, both the accuracy and the model size would be considered since the Software should be accessible to everyone
- Also, the model will be used in both code generation and explanation tasks as well
- The UI will be used to get the user intention like if it's a generation or an explanation like GitHub copilot

Week 5: First Evaluation Preparation

- Implement basic FastAPI endpoints to serve the initial model.
- Document the initial training process and API endpoints.
- Prepare for the first evaluation.

Week 6: Model Abstraction Layer Implementation

- Design and implement the model abstraction layer to ensure model-agnostic behavior.
- For this, the dataset that we have used will be open-sourced, so anyone can use any model after fine-tuning a model
- Integrate the LLM into the abstraction layer.

Week 7: ANN Implementation and Data Retrieval Optimization

- Implement ANN algorithms to efficiently retrieve relevant data(Planning to use LangChain and FAISS).
- The RAG pipeline will be implemented using long-chain, FAISS, Ollama, and FastAPI
- Optimize data retrieval processes for improved performance.

https://python.langchain.com/docs/integrations/vectorstores/faiss/

https://www.kaggle.com/code/akashmathur2212/demystifying-faiss-vector-indexing-and-ann

Week 8: Second Evaluation Preparation

- Refine the FastAPI endpoints and improve error handling.
- Implement hallucination mitigation techniques.
- Prepare for the second evaluation.

Week 9-10: Hallucination Mitigation and Refinement

- Implement and refine techniques to minimize LLM hallucinations.
- Conduct thorough testing and debugging.

Week 11: Documentation and Finalization

- Finalize documentation, including technical guides and user documentation.
- Prepare the project for long-term maintainability.

Week 12: Final Submission

- Finalize the projects and get approval from the mentors

Hours per Week

I plan to dedicate approximately 30 hours per week to this project.

Progress Reporting:

- Regular weekly updates through GitHub pull requests and issues.
- Emailing my mentor to discuss progress and address any challenges.
- Detailed progress reports before each evaluation.

4. Post-GSoC Plans

I am committed to continuing my contributions to Sugar Labs after GSoC. I plan to:

- Maintain and improve the AI-powered Music Blocks code generation system.
- Contribute to other Sugar Labs projects.
- Actively participate in the Sugar Labs community.

And even if I don’t get selected I would be happy to take part in the Sugar Labs Community