ResoluteAI Software
SaaS | Solutions | Services | Support

Greetings from ResoluteAI.in!

Thank you for your interest in our internship opening. As a next step in the screening process, you are required to complete the below-mentioned assignment.

Role: AI Engineer Intern
Duration: 48hrs.
Data Link: Please Click Here to access data of all the below tasks

Choose any two from the options below.
One from Task 1 to Task 3
Second from Task 4 to Task 7

Mandatory Tasks: Tasks should be mandatorily completed.

Task 1: OCR
Complexity: Easy

User Story: As a user, I should provide a path to the image, and the program should display the text from the image. (You are free to use open-source models and codes, but please ensure there is no complete copy-paste done)

Task 2: Document Extraction
Complexity: Medium

User Story: As a user, I should provide a path of PDF, and the program should display the text from the Pdf. (you are free to use open-source models and codes, but please ensure that there is no complete copy-paste done)

Hints: Table information should be in the List_items section, other information can be taken as headers of JSON file. Submit the collab-notebook/code file. JSON file

Task 3: Crowd Detection
Complexity: Medium

User-Story: Train an AI model to detect persons in a video & further analyze the detected data to identify crowds. This involves leveraging pre-trained object detection models and applying custom logic to detect and log crowd events.

Crowd Detection Logic hints :

- A crowd is defined as three or more persons standing close to each other for 10 consecutive frames.
- Identify groups of three or more persons standing close together in a frame.
- Check if the identified groups persist for at least 10 consecutive frames.

- If a crowd is detected, log the frame number and the count of persons in the crowd.
- Save the results in a CSV file with the Frame Number, Person Count in Crowd

Task 4: Create a Retrieval Augmented Generation (RAG) Application in Streamlit
Complexity: Medium

User-Story: Build an interactive Streamlit application where users can upload multiple documents and chat with those documents using RAG.

Hints:

- Implement a file uploader to allow users to upload multiple documents (PDF, DOCX, TXT).

- Parse and preprocess the uploaded documents.
- Use a Large language model and a retrieval mechanism to augment the generation.
- Integrate with a document retrieval library to fetch relevant passages from the uploaded documents.
- Build a chat interface where users can ask questions.
- Display responses generated by the RAG model using the retrieved document passages.
- Test the application with various documents and queries.
- Submit the screen recording of the working application.

Task 5: Finetune Llama2-8B or Llama3-7B Model on a Custom Dataset
Complexity: Medium

User Story: Fine-tune a large language model on a specific dataset to adapt it for a particular task.

Hints:

- Setup Google Colab(Gpu) with all required libraries like Hugging Face Transformers, PyTorch, etc.
- Collect and preprocess the custom dataset, Format the dataset in a way suitable for fine-tuning (e.g., JSON, CSV).
- Load Llama2-8B or Llama3-7B using Hugging Face Transformers and prepare the model for fine-tuning.
- Write a script to fine-tune the model on your dataset & adjust hyperparameters(learning rate,batch size, epochs) as needed.
- Evaluate the fine-tuned model on a validation set.
- Analyze performance improvements and possible overfitting.
- Save the fine-tuned model and push it to HuggingFace..
- Document the fine-tuning process, including the dataset, parameters, and results.
- Submit the Colab Notebook and link to the model deployed on cloud (Hugging Face)

Task 6: Create a Custom Agent Using LangChain that can generate code and execute It
Complexity: Hard

User Story: Develop a custom agent using LangChain to generate and execute code based on user prompts.

Hints:

- Define the scope and capabilities of the agent (e.g., languages it can code in, types of tasks it can handle).
- Integrate a language model (e.g., GPT-3) with LangChain to handle code generation.
- Ensure the agent can run and test code safely.
- Write the logic for the agent to understand prompts, generate code, and execute it.
- Test the agent with various coding tasks and prompts.

Task 7: Create a Visualization Pipeline Using LLMs
Complexity: Hard

User-Story: Create a visualization pipeline using any LLM of your liking, Create a streamlit application where the user will be able to upload his structured data, which could be CSV or EXCEL, and ask questions from that file. LLM should perform the required analysis to answer the user's question. Make your LLM generate code based on the user's question and then execute it. (Easy way to do it is to use the agents provided by the langchain for it. If you use it, you will need to explain how the agent works in the background.)

Hints:

- Implement a file uploader to allow users to upload CSV or Excel files, and parse and preprocess the uploaded data.
- Use an LLM (e.g., GPT-3) to generate code for data analysis based on user questions.
- Implement LangChain agents to handle code generation and execution.
- Execute the generated code securely.
- Display the results and visualizations in the Streamlit app.
- Test the application with various datasets and questions.
- Validate the accuracy and relevance of the analysis.

Submission:

- Send a screen recorded video of the user story or upload it into your Google Drive and share the link (please ensure to rename the video to your full name before sending it)
- Once approved we will ask for the code
- Please zip the video before sending
- **Rename structure: TASKNUMBER_FULLNAME**

Deadline:

48 hours after receival of the Assignment.

Note: Approach will be given More Value in assessment.