CalQuity

Intern Assignment - AI Engineer

About Us

CalQuity's mission is to provide investors and analysts with state-of-the-art research tools that simplify and expedite investment decisions. We currently have a 4 member team, and are looking to hire an AI Engineer to accelerate our product development.

We have been accelerated by premier startup programs like Razorpay, Microsoft among others. Our first product is on the market, with 2 new products coming out soon. You can find out more about us on our website - calquity.com

For the role of AI Engineering Intern, we require candidates to have a basic understanding of backend development in Python (preferably with some experience in developing LLM/Agentic Applications), and the ability to learn fast. The internship is ideal for students in college, as we are flexible with work timings.

From an assignment perspective, you need to:

1. **Choose one** of the three given tasks.
2. **Create a Github repository** containing your response.
3. **Include your name and email address** in the repository.
4. **Submit a brief video (3-4 minutes)** - Give a quick introduction about yourself, along with a walkthrough of the code, process you took to get the required results and a demo of the output. No need to go into too much detail. Intro + Explanation should be < 5 minutes.
5. **Include the instructions** on how I can replicate your output.

Task 1: Basic Agent with Tool

Objective

The objective of this task is to construct a basic agent and equip it with a custom tool. The agent should effectively utilize the tool to perform a specified function. You can use the agentic framework of your choice (LangGraph, AutoGen, CrewAI etc), or go ahead without choosing a

---

specialized framework. We recommend using AutoGen/LangGraph for those without any specific preference.

We understand if you haven't used any of the above mentioned frameworks before. This task is meant for you to explore and get accustomed to Agentic Frameworks, and demonstrate your ability to pick up new frameworks quickly.

Methodology

1. Agent Development

- Build a rudimentary multi-agent application capable of executing predefined tasks.

2. Tool Creation

- Develop a simple tool using the yfinance API to fetch financial data such as stock prices, and visualize it.

3. Integration and Demonstration

- Integrate the tool with the agent.
- Provide specific instructions to the agent and showcase its ability to use the tool effectively.

Deliverables

- Think of a particular use case, and implement a basic agent for it. Clearly state what your agent is capable of doing.
- A functional tool built using the yfinance API. Feel free to use any other API that better suits your use case.
- A demonstration of the agent's ability to utilize the tool effectively. Feel free to use as many tools as you like. Refrain from using existing implementations available on the internet, We want to see something cool!

Task 2: Embedding Model Comparison and RAG Pipeline

Objective

The primary goal of this task is to evaluate the answer quality of two embedding models, Cohere AI and Voyage AI, within a Retrieval-Augmented Generation (RAG) pipeline.

Methodology

1. RAG Pipeline Implementation

- Construct a RAG pipeline that utilizes both Cohere AI and Voyage AI embedding models.
- Ensure the pipeline includes document retrieval, embedding generation, and answer synthesis. You don't need to make the pipeline complicated, you can use a template pipeline.

2. Observability and Evaluation Metrics

- Integrate Athina AI and/or Weights & Biases Weave to track and visualize key metrics.
- Monitor embedding quality, retrieval performance, and overall answer quality.

Deliverables

- A well-documented RAG pipeline implementation.
- A concise evaluation framework for answer quality.
- A comparative analysis of Cohere AI and Voyage AI performance.
- Insights derived from observability and evaluation metrics. You can use the preset Metrics in Athina + Traditional Metrics to do the comparative analysis.
- A final verdict on which embedding is working better, and why (with metrics + examples)

Dataset for documents, and questions to test on. Here you will find the correct answer as well, if you want to run benchmarks on correctness. PatronusAI/financebench · Datasets at Hugging Face

Task 3: Hybrid Search in PGVector

Objective

Implement a hybrid search application using PGVector and compare the performance of dense and dense+sparse search strategies.

Methodology

1. PGVector Setup

- Configure a PostgreSQL database with the PGVector extension.
- Consider using Supabase for database hosting.

2. Hybrid Search Application

- Develop a hybrid search application leveraging both dense and sparse vectors.
- Implement appropriate indexing and query strategies for both search approaches.

3. Dense vs Dense+Sparse Comparison

- Use a diverse set of queries and datasets to ensure a comprehensive comparison. The idea is to evaluate how sparse vectors affect the quality of RAG systems.

Deliverables

- A fully functional hybrid search application using PGVector.
- A detailed experimental setup for comparing dense and dense+sparse search.
- A comparative analysis of search performance.
- Insights and recommendations based on the findings.

You can use Athina to setup LLM as a judge comparisons, side by side for both setups. Try to run at least 20 questions to identify differences. For questions and sample data, you can use https://huggingface.co/datasets/PatronusAI/financebench.