Link's To the most complex Python Code And Database:- https://github.com/Anon675

1)Self Rating (LLM, Deep Learning, AI, Machine Learning)

A) LLM-B

- I can work productive with large language models like using extensive frameworks Api, I know some core concepts like tokenization's, embeddings or prompt design, RAG.
- I can build a LLM based applications and integrate them with external systems.
- Aware of practical limitations like hallucination context window constraints and prompt sensitivity.
- Experience handling multi-turn interactions when managing context outside the model.

B) Deep Learning-B

- I'm familiar with CNN's, RNNS and Transformers at an architectural level.

- Understand GPU VS CPU limitations at a practical level.
- I understand the role of regularization techniques such as dropout and early stopping.
- Can debug common issues like loss not decreasing or unstable training.

C) Machine Learning- B

- Comfortable handling the classical machine learning lifecycle end to end from data cleaning to model evaluation.
- I have strong understanding of supervised learning algorithms and when to use them.
- Experience working with noisy or imperfect real-world data.
- I understand the bias-variance trade-off and how it affects model performance.

D) AI-A

- I understand the importance of data quality over model complexity.
- I have experience in integrating ML models with traditional backend services.
- I understand tradeoffs between accuracy latency and cost.

- I’m comfortable combining ML outputs with rules or heuristics when appropriate.

2) Key architectural components to create chart based on LLM

The architectural components that are involved are is as follows: -

Step-1 User interface layer

- A client interface is used only to collect user input and display responses.
- No AI or decision logic is kept at this layer to avoid tight coupling.

Step-2 Back-end service

- A back-end service receives user requests and acts as the main control point.
- It handles authentication, request validation and basic rate limiting.
- Is the layer that decides whether the request needs simple generation or external data resources.

Step-3 Context handling

- In this conversation history is stored outside the model.

- Text is summarized or dropped so that it stays within the limits that are assigned to the token.
- Message requests are sent for each request only when the past messages are relevant.

Step-4 Knowledge lookup

- The user's query is not matched by keywords it is matched semantically.
- The system retrieves relevant information from the stored document when the when the question is of specific domain.
- To reduce hallucination in the system, retrieve content is added to the prompt.

Step-5 Prompt assembly

- To avoid exceeding model limit prompt size is controlled.
- User input instructions and Retief contexts are combined together to make the final prompt.
- Instructions and user contexts are clearly separated.

Step-6 Model call

- The output is not rated as a guaranteed correct answer it is treated as a draft.
- The prompt is through an API to the LLM.
- Either accuracy or creativity is used to choose the generation parameters.

Step-7 Output checks and feedback

- Before returning the response to the user big guardrails are applied to check the out and for obvious errors or formatting issues.
- Logs are reviewed to improve prompts and retrieval logic so that the changes are made incrementally without retraining the model.

3) Vector database

- A vector database is a type of specialized database that is designed to store index and search high dimensional vector representations of data known as embeddings.
- It is a type of database that uses similarity search techniques such as cosine similarity or Euclidean distance find the items that are semantically or visually similar i.e. “instead of asking “does this match exactly?” in this the system asks “how close is this in meaning?””
- Now let us understand this with an example: -
    - Let us suppose we have to build a chat bot to answer questions from internal company documentation
    - In which the documents are split into smaller chunks and then each chunk is converted into embedding and stored, so when a user asks relevant document

sections are retrieved from semantic similarity and the retrieved content is added to the LLM prompt to generate a grounded response.

- With this approach the answers are aligned with the real data and the hallucinations are reduced.
- There are many types of vector databases that are used for this kind of system such as FAISS, Pinecone, Milvus, Chroma.
- For this internal type of documentation chat bot FAISS would be a reasonable initial choice because the decision is driven by practical constants rather than feature comparisons i.e. the data is internal and does not need to be sent to an external management service, latency requirements are manageable without horizontal scaling, System can be deployed as a single service without complex distributed infrastructure.
- Other vector databases like Pine Cone can also be used but for this specific use case those benefits are not immediately required like scale availability and multi-tenant access becomes critical and introducing a managed service early would add cost and operational dependencies like without any clear benefit at the current scale but it can allow switching

to a managed solution in future if usage or data size grows.