Problem statement:

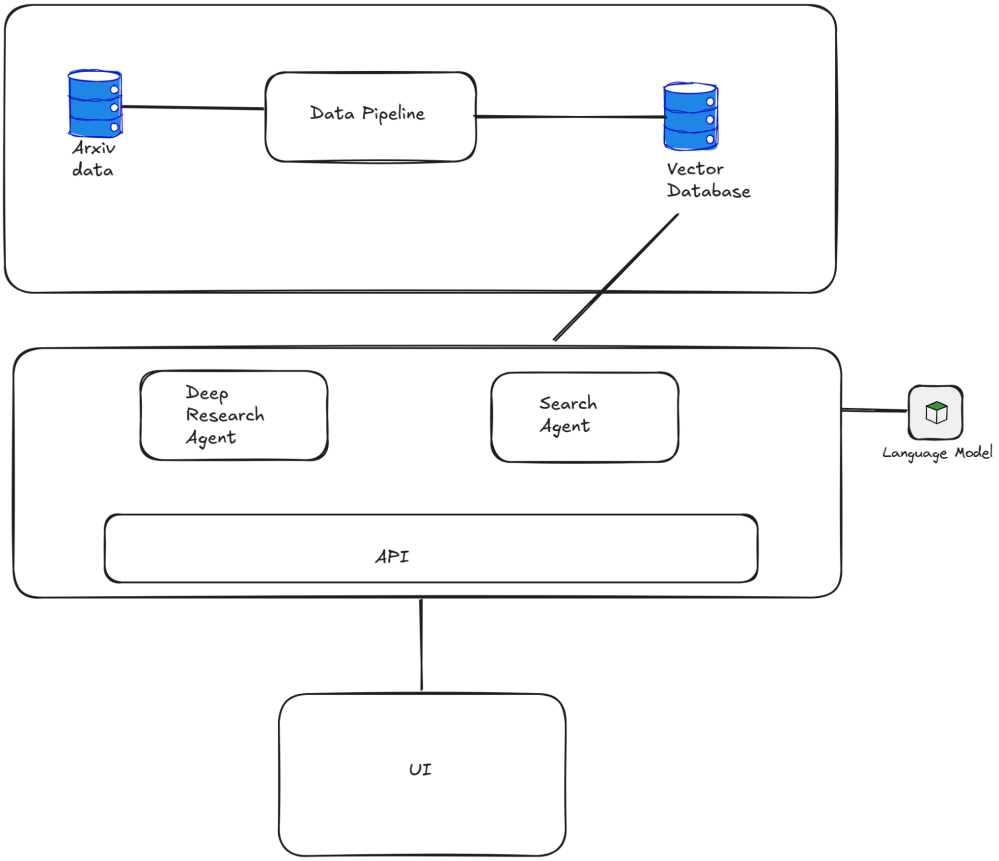
Building different versions of agents on top of Arxiv data

- Simple chatbot
- Deep Research agent

Data Sources

https://www.kaggle.com/datasets/Cornell-University/arxiv

Technical Blocks



Data Pipeline : Simple python script that can convert the data from the source into embeddings and store in a vector database.
Deep Research Agent : An agent that will combine data from vector database and Browser to get detailed research on user topic.
Search Agent: A chatbot on top of vector database.
API : Simple API layer for the UI to enable this communication.
UI : A simple interface to enable user interaction with the system.

Libraries and Frameworks

- **Language** : Python
- **Agent framework** :
    - langgraph for agents
    - hugging face for embeddings
    - fast api for api
    - gradio for frontend.
    - Pytest for unit tests
    - Tools - MCP (if required)
    - Dependency management - uv
- **Vector database** : Chroma db
- **Language model** : Qwen/Qwen2.5-Omni-7B

Data Schema Understanding

- List of available columns
    - id
    - submitter
    - authors
    - title
    - comments
    - journal-ref
    - doi
    - report-no
    - categories
    - license
    - abstract
    - versions
    - update_date
    - Authors_parsed
- Columns indexed
    - Title - embedding
    - Abstract - embedding
    - Authors - Meta strings
- Search criteria
    - If the query contains the author's name only then we will consider.
    - Otherwise, simple retrieval based on HNSW
    - Enhancements to the search can be done later.

Evaluation Plan

- How will you know your chatbot is performing well?
    - Manual review -
- How will you know your Deepresearch is performing well?
    - Manual review -

Agent Logic

- DeepResearch agent
    - The agent developed using langgraph will be given a browser and a set of websites to search for and come back with an answer.
- Tools
    - MCP based tools wherever it is necessary
- Search Agent
    - Simple memory to keep track of working context
    - The memory will be part of Chroma db itself

Model Selection Validation

- Keeping API costs in mind everything will be in local for now.

Agent Plan & Evaluation Addendum

- Deep Research Agent Structure
    - The Deep Research Agent will be built using LangGraph, with modular tools for:
        - Retrieving internal research (from vector DB)
        - Browsing the web (via an MCP browser tool or SerpAPI)
        - Synthesizing answers via LLM
- **Agent Nodes:**
    - **Input Parser Node** – Decides whether external data is needed.
    - **Retriever Node** – Fetches top-k results from ChromaDB.
    - **Browser/External Search Node** – Uses external web tools to enrich context.
    - **Synthesizer Node** – Combines internal + external sources using Qwen model.

LangGraph Flow Logic:

- If query is specific to ArXiv:
    - `→ Retrieve from Vector DB → Synthesize`
- If broader:
    - `→ Retrieve from Vector DB → Search Web → Synthesize`

Automatic Evaluation Plan

- **For Search Agent (Retrieval):**
    - **Recall@k**: Check if expected documents are among top-k retrieved.
    - **Embedding Match Score**: Log cosine similarity or dot-product scores.
- **For Deep Research Agent (Generative):**
    - **Faithfulness**: Does the answer include references (e.g., paper title, URLs)?
    - **Coverage**: Use BERTScore or ROUGE-L against a small reference set.
    - **LLM-as-a-Judge**: Ask an LLM, “Does this answer address the research question accurately and completely?”