Market Researcher: Supply Detection

Recently Zudio has been a big success in our economy. Zudio is a clothing brand for men and women and is part of the Tata group. The success behind Zudio is mainly attributed to the business idea of creating supply where it was not present. The firm found that there were no companies supplying garments with good designs at affordable prices. So they decided to take up this space in the market and since they were a monopoly in this segment the brand was a huge success in our country. I would like to create a LLM to find out any spaces like these available in the market which companies can take up and make huge profits from it.

So for this project I have created a prototype for looking into only the mobile phones of our country. Now the person has to query the LLM and find out any spaces like this manually. But later with further insights we can make the LLM find out this on its own and also implement for multiple products in the market.

Architecture and Tools Used

At the heart of this agent is the **LangChain framework**, which provides modular building blocks for combining language models, vector stores, and prompts. The project uses **Ollama** to serve LLaMA locally, for vector storage and retrieval, **ChromaDB** is used, and the embedding model mxbai-embed-large converts mobile data rows into semantic vectors.

The mobile dataset is first parsed using pandas, chunked into readable text rows, and added to Chroma. LangChain's retriever performs similarity search based on user queries, which is then passed into a prompt template alongside the query and processed by LLaMA.

The backend is served through **FastAPI**, providing an endpoint /run-agent that accepts user questions and returns AI-generated responses. A simple **Streamlit frontend** offers a clean UI where users can type questions and receive answers in real-time.

Evaluation Strategy

The model can be evaluated based on the retrieval of the required information and the output the LLM generates. The retrieval part of the model can be tested partly automatically based on the condition required and the results from the retrieval method. However the inference the LLM makes from this data needs to be tested manually.

Knowledge Graphs and Agentic RAG

While this prototype primarily uses vector-based retrieval, it can be extended to incorporate a **Knowledge Graph** to represent structured relationships — such as brand → model → specs → price. Using **LangGraph**, the agent could perform **multi-hop reasoning**, enabling it to answer layered queries like "List phones by Realme launched after 2022 with fast charging and

AMOLED displays.” in a better way. Graph-based RAG would improve both interpretability and retrieval precision in such cases and would also increase the explainability of the model.

Adaptation and Improvement Loop

The agent can be extended to include a **feedback mechanism** where users rate the response quality. Over time, this feedback can be used to re-rank retrieved documents, refine prompts, or fine-tune retrieval weights. Integration with LangGraph memory nodes could also allow context-aware follow-up queries.

Challenges Faced

In the first implementation of the LLM I had tried to extract the data directly from the csv file but it was not able to recognize the attribute of the entry. So I wrote a function to convert the data extracted from the csv file to create chunks of data that can better represent the data for easier understanding of the LLMs.

Later I also faced some issues for the FastAPI and streamlit segment of the project which was solved by looking into similar issues raised by other people from the internet.