Solution Report for S2T Interview

N K Vamsi Krishna

December 12, 2024

Introduction

This report details the solution design, challenges faced, and strategies employed to address those challenges for the **NER & RAG App**. The project integrates backend AI-based services with a frontend visualization tool to provide a seamless user experience for **entity and relationship extraction** and **RAG (Retrieval-Augmented Generation)** functionalities.

Solution Design

Core Approach

- The backend comprises **microservices** using Flask, with a gateway service (main service) handling AI tasks and coordinating CRUD operations for entities and relationships. REST APIs facilitate communication between services, and JSON responses ensure structured data transfer.
- The frontend, built with **Next.js**, visualizes extracted entities and relationships using the `React Force Graph` library. It also includes a **RAG Q&A interface** for interactive querying.

Service Communication

The backend employs REST APIs using Flask for inter-service communication. Each service exposes its functionality through well-defined routes, ensuring modularity and scalability. **JSON** is used as the response format for consistent data exchange.

Frontend-Backend Integration

The frontend communicates with the backend using `Axios` to perform API requests. The data flow is simple, with specific API routes for graph rendering and Q&A functionalities. Advanced mechanisms like polling were not required due to the system's current scope.

Challenges Faced

Hardware Constraints

Running local LLMs (e.g., **Phi3** and **Llama 3.2**) on an M1 Mac with 8GB RAM posed significant challenges:

- **Memory bottlenecks:** Due to simultaneous execution of multiple Docker containers.
- **Lack of GPU acceleration:** Slower inference for LLM tasks.

These limitations necessitated careful resource management.

Model Limitations

Smaller models like **Phi3** and **Llama 3.2** exhibited slow inference times (up to 4 minutes) and occasional hallucinations. To address this:

- A `check_relation_response` function validated outputs by ensuring source and target IDs matched known entities.
- **Programmatic filtering** was employed to remove invalid relationships.

Frontend Visualization

The `React Force Graph` library performed well for the project's scale. It provided built-in functionalities such as **zoom, pan, and physics-based interactions**, enhancing user experience.

Environment Configuration

No major issues were encountered with `.env` variables. The OpenAI and Pinecone APIs were smoothly integrated into the backend services.

Problem-Solving Strategies

Debugging Model Outputs

Structured outputs from LLMs were validated using **Pydantic schemas**. The solution leveraged Ollama's structured outputs and OpenAI's `response_format` method for reliable parsing. This approach ensured accurate and consistent data extraction.

Error Handling

The backend implemented robust error-handling mechanisms, including:

- **Try-catch blocks:** For handling exceptions gracefully.
- **Custom error messages:** For API failures and model errors.

User Experience Enhancements

To provide a responsive and intuitive interface:

- **ShadCN components** and **Tailwind CSS** were used for styling.
- The `React Force Graph` library provided animations, node/link coloring, and interaction features.
- A **legend** and a `fit view` button ensured ease of navigation.
- **Info modals** displayed detailed information about nodes and links.

Frontend Visualizations

NER Graph

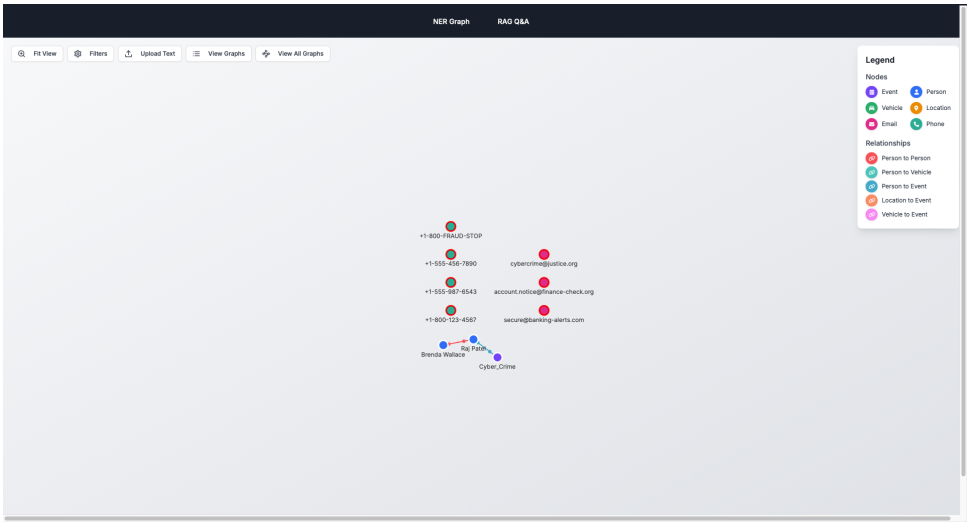

Figure 1: **NER Graph** showing entities and relationships.

RAG Q&A Interface

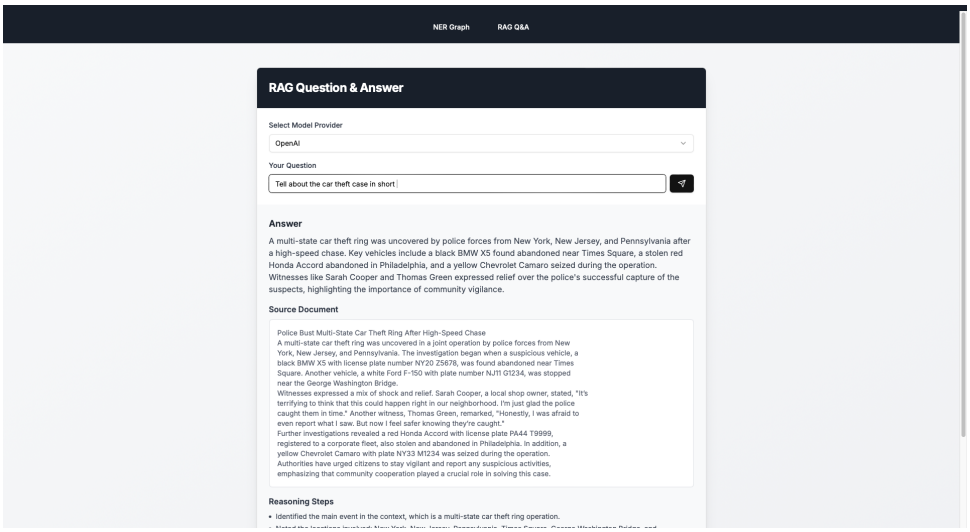

Figure 2: **RAG Q&A interface** for querying extracted entities and relationships.

Conclusion

The **NER & RAG App** successfully integrates backend AI capabilities with a modern frontend for visualizing entities and relationships. Despite hardware and model limitations, the challenges were effectively addressed through:

- Programmatic validation,
- Structured outputs, and
- Robust design principles.

The solution demonstrates **scalability, user-centric design,** and **efficient resource management.**