Datathon Challenge

Designing a RAG Agent on Raw HTML Data

Gieni by Orderfox X *ETH Analytics Club*

1. Introduction

In today's AI landscape, Agentic AI is considered a major technology advancement unlocking significant value. A key part of Agentic AI are Retrieval-Augmented Generation (RAG) systems, which offer a powerful way to combine large language models with external knowledge. In this challenge, your task is to design an end-to-end RAG pipeline using a real-world dataset composed of unstructured text extracted from HTML.

Your mission: **transform messy web data into structured knowledge and build an intelligent agent on top of it**.

2. Background

Gieni AI is a groundbreaking B2B Agent focused on generating deep market insights with its unique research reports and extensive company data. A key application of our data is the analysis of supply chains.

Gieni gathers data from multiple sources, including open databases like EDGAR and Eurostat. One of our most technically challenging datasets is website data used for company profiling. You will use a sample of this rich dataset to build a *supply chain analysis agent* capable of answering questions about companies across multiple categories.

3. The Challenge

You will receive a subset of raw HTML documents from our production data scraping pipeline. The challenge is to develop a working prototype of a RAG agent that can answer user questions based on this dataset. The project is open-ended — your team will decide how to parse, store, retrieve, and generate.

A user of your agent will typically be a supply chain director who will be overseeing the logistics and procurement of key vendors for their business. Supply chain experts are often concerned with risks to their supply chain, such as capacity risk or geopolitical risk. Additionally, a supply chain expert might be concerned with opening new markets or new product categories and need to source suppliers in a new location. Your agent will address these questions directly.

A user of your agent should be able to ask questions across 6 main categories: technology, services, materials, products, industries and regions. For example, a user might ask "Which companies in Southern Italy produce aluminum auto components?" or "I am concerned about geopolitical risks in my supply chain, in which region should I look for alternative specialty chemical suppliers?" These categories and kinds of questions can be helpful in defining the scope of the Agent's knowledge. These questions are an example of what could be answered by an ideal chatbot. Building agents that can retrieve, reason and generate output that matches intent are the biggest challenges in building RAG systems.

You are free to define your architecture and tech stack. Creativity and engineering quality are key.

4. End User Persona

Supply Chain Director

Our primary end user is a Supply Chain Director responsible for overseeing vendor relationships, logistics, and procurement. They need actionable insights to identify risks (such as capacity or company risks), source new suppliers, and explore alternative markets. Their decisions directly impact operational efficiency and cost management, so they value quick, reliable, and contextually rich answers that can support strategic planning.

5. Your Tasks

a. **Parsing & Preprocessing**

- Extract relevant content from raw HTML (e.g. product details, text, metadata).
- Clean, chunk, and format it for retrieval.

b. **Knowledge Base Design**

- Store the parsed data in a way that supports retrieval (some common solutions include vector DB, graph DB, relational (SQL) DB, NoSQL, and others — your choice).
- Justify your design choices.

c. **Retrieval System**

- Implement a retrieval pipeline to find the most relevant data given a user query.

- You can use LLMs, semantic search, SQL queries or any other retrieval methods in your design – be creative and innovative.

d. **Generation Agent**

- Combine retrieval results with an LLM (e.g. OpenAI GPT-4. via API key we provide) to answer questions.
- Prompt design and engineering, and memory structure is up to you.
- Justify your designs and parameterization. Consider key LLM parameters such as *temperature*, *instructions*, *reasoning*, etc.
- See OpenAI API reference here.

e. **Bonus Points (only consider these if you have time)**

- Handle ambiguity, hallucination, or low-confidence answers gracefully
- Include a minimal frontend, CLI or interface to interact with the agent
- Propose metrics for evaluating response quality
- Handel queries that require reasoning such as counting, decision making etc., for example “How many dental practices are there in Zurich?”
- Having a conversational engagement rather than simply inputting a query and getting the result. Allow a back and forth with a user before the RAG event is triggered.

6. Deliverables

- A short report (max. ***1-2*** pages) describing your system design, tools used, and what tradeoffs you made
- A GitHub link to your codebase
- A live demo or screen-recorded walkthrough
- A final ppt presentation (PDF) summarizing your architecture, challenges, and insights.

7. Evaluation Criteria

(see attached evaluation document)

Your solution will be evaluated based on:

- **Effectiveness**: how well the agent retrieves and answers based on the data
- **System design**: architecture decisions, explainability, flexibility
- **Cost:** Choose your OpenAI model carefully taking into account budget limitations ($40) and the cost of input and output tokens
- **Code quality**: structure, clarity, and reproducibility
- **Creativity & insights**: novel approaches, extra features, thoughtful design, new technologies.
- **Presentation**: clarity and insightfulness of your final presentation

8. Tools Provided

- Sample HTML dataset in json format
- OpenAI API key
    - Usage limits apply: $40
    - A pricing list for the different models can be found here
    - No search models, and general websearch functionality in your agent is allowed, the goal is to build a RAG agent not a search bot!
- Jupyter Notebook
- Slack channel for Q&A with our team

(see attached document for suggested tools - if needed)

9. Important Notes

- The challenge is open-ended. You won't be penalized for what you don't build — but rewarded for what you do, and how well you explain it.
- Think like a startup team: limited time, high impact, iterate fast.
- Exclude web search models! This is not the goal of this challenge.