UserGems Technical Challenge Report

Sina Ahmadi

April 2025

1. Overview

This proof-of-concept project for UserGems demonstrates an end-to-end Retrieval-Augmented Generation (RAG) system that extracts and reasons over publicly available company information from the web. The goal was to build a working prototype within an 8-hour window that showcases scraping, enrichment, retrieval, and LLM-assisted answering.

The full codebase, including all scripts, pipeline stages, and example queries, is available on GitHub at: `github.com/sinaaat/usergems-challenge`

2. Technology Choices

2.1. FAISS for Vector Storage

I selected FAISS (Facebook AI Similarity Search) as the vector database due to its performance and ease of integration with LangChain. It runs efficiently on local machines, doesn't require complex setup, and provides reliable nearest-neighbor retrieval for semantic search.

2.2. LangChain for Chunking and RAG

LangChain was used to manage chunking, document construction, and to simplify integration between vector stores and language models. It provided a clean abstraction for creating chains and pipelines, which allowed me to iterate quickly without writing boilerplate for embeddings, retrievers, or QA logic.

2.3. OpenAI GPT-4o for Metadata and Answers

I used GPT-4o to extract metadata from text (e.g., business model, pricing info) and to generate context-grounded answers. GPT-4o's reasoning ability and reliability in generating structured outputs made it ideal for these tasks.

3. System Design

Step 1. **Scraping**: Websites were loaded from a CSV and scraped using `requests` and `trafilatura`, covering both homepages and common subpages (e.g., `/about`, `/services`).

Step 2. **Enrichment**: Website content was augmented with subpage data to improve context richness.

Step 3. **Metadata Extraction**: GPT-4o was used to extract structured metadata:

- `business_model`: B2B / B2C / both
- `pricing_mentioned`: boolean

- `price_hint`: textual pricing information

Step 4. **Chunking and Embedding**: Content was split into overlapping chunks (800 chars) and embedded using OpenAI Embeddings. Chunks were stored in FAISS with attached metadata.

Step 5. **Question Answering**:

- `query.py`: basic semantic search + GPT answer
- `query_rerank.py`: retrieves 10 chunks, reranks with GPT, and answers from the top 3

4. Examples and Evaluation

Open-Ended Query

User Input: *Which companies offer affordable consulting services?*
Baseline Result (query.py):

- **Answer:** Based on the provided context, KMC Consulting Solutions Inc. and Carriage House Consulting offer consulting services that focus on cost-effective and efficient solutions to help businesses grow and thrive. They emphasize understanding business operations, creating implementation strategies, and providing guidance for success.
- **Chunks Used:**
  - KMC Consulting Solutions Inc. (chunk 0)
  - Carriage House Consulting, LLC (chunks 2 and 5)
  - KMC Consulting Solutions Inc. (chunk 1)
  - Datalux (chunk 2) – this was irrelevant to the query

Reranked Result (query_rerank.py):

- **Answer:** The companies mentioned in the context that offer affordable consulting services are KMC Consulting Solutions Inc. and Carriage House Consulting. Both companies focus on providing cost-effective solutions to help businesses grow and improve their operations.
- **Chunks Used:**
  - KMC Consulting Solutions Inc. (chunks 0 and 1)
  - Carriage House Consulting, LLC (chunk 5)

Observation: The reranking version filtered out the unrelated Datalux chunk and provided a more focused and concise response by selecting the top 3 most relevant chunks using GPT-4o.

Structured Query Limitation

User Input: *Give me the list of all companies that have a product less than $20,000.*
Output: GPT-4o responded: *"I'm sorry, but I don't have access to a list of companies with products priced under $20,000."*
Chunks Retrieved:

- Futurae Diamonds (chunk 3) – mentions free shipping over $500
- Disability Compliance Solutions LLC (chunk 4) – mentions $1.25/page printing

- Industry Knowledge Graph (chunks 0 and 2) – unrelated
- Hope for Youth and Families Foundation (chunk 5) – irrelevant

Analysis: This query type requires structured numeric filtering rather than semantic similarity. GPT's honest refusal to hallucinate data reflects a strength in system reliability, but it also highlights the need for numeric metadata parsing and filtering (e.g., `estimated_price < 20000`).

5. Limitations and Improvement Areas

While the core RAG system performs well for open-ended semantic queries, several limitations emerged when addressing more structured or business-specific tasks. These limitations point toward the natural boundaries of semantic similarity and the need for richer, more structured data extraction.

- **Sparse Pricing Information**: Few websites disclosed pricing clearly. This limited the effectiveness of queries requiring numeric thresholds or pricing comparisons.
- **No Structured Price Filtering**: Although I extracted a `price_hint` field using GPT, there was no numeric normalization (e.g., \$19,000 $\rightarrow$ 19000) for filtering. This limits support for queries like "products under \$20,000."
- **Limitations of Semantic Search Alone**: Not all tasks can be solved with similarity-based retrieval. Structured queries require relational or rule-based logic that semantic embeddings alone cannot capture.
- **Potential of Deeper Scraping (Partially Integrated)**: I integrated subpage enrichment logic to fetch deeper content like pricing or product pages. While this lays the groundwork for richer answers, it's not yet efficient or fully automated. In practice, this approach could address many current limitations — but it requires more time-intensive crawling logic, smarter link following, or sitemap parsing to become production-ready.
- **No Hybrid Filtering or UI**: There is currently no hybrid search strategy that combines metadata filtering with semantic retrieval. Similarly, no user interface was implemented to make this system easily testable by non-technical users.

6. Next Steps (Not Implemented)

- Enrich the metadata
- Add structured filters (e.g., `pricing_mentioned=True`)
- Introduce Streamlit app for interactive exploration
- Evaluate performance with standard QA metrics (precision@k, etc.)

7. Conclusion

This project successfully demonstrates a working pipeline for Retrieval-Augmented Generation (RAG) using publicly available company data. The system combines semantic chunking, metadata extraction, vector-based retrieval, and GPT-4o reasoning to answer natural language queries.

The project shows reasonable performance on open-ended, descriptive questions and highlights the limitations of semantic search when structured filtering is required. Reranking strategies proved effective in improving precision, while metadata extraction lays the foundation for future structured reasoning and filtering.

Although some components such as structured price filtering and deep enrichment were not fully developed due to time constraints, the system is designed in a modular and extensible way to support future enhancement. Overall, the prototype provides a clear and adaptable architecture for intelligent company profiling and question answering at scale.