Parrandata - Datathon 2025 - Orderfox

Carlos Hurtado, Marc de Miguel, Alex Martí and Eias Abad Rocamora

Data Preprocessing

The initial dataset is composed of **17.31 GB** of data. In order to deal with a smaller and cleaner dataset, we decide to remove the subpages that are not relevant for our task, including .css files, videos, privacy-policy pages and language-duplicated pages.

- **Step 1:** Raw files (**17.31 GB** to **8.08 GB**)
  In this step, we remove files with the following extensions:
  ".css", ".png", ".jpg",".jpeg", ".gif", ".mp4", ".pdf"
  This way, we avoid computing embeddings of non-text files that will not be useful.
- **Step 2:** Language duplicates and irrelevant pages (**8.08 GB** to **7.82 GB**)
  In this step we check if the webpage is a language-specific page, e.g., /es/ or /fr/. Additionally, we check if the subpage url contains keywords like: "/privacy-policy", "/terms-of-use", etc.
- **Step 3:** Large files (**7.82 GB** to **4.81 GB**)
  Lastly, we remove some very large files that were not removed in the first step. Sometimes, some very large and high quality pdfs are present in a link without the .pdf extension, e.g., "http://nov.com/about/sustainability/report/2023". In order to filter out these files, we remove pages longer than 200k characters.

Embedding Creation and Vector Database:

We have used OpenAI's text-embedding-3-small model to encode the text. Due to rate limits, we only embed the homepage of each company. We store the embeddings in a ChromaDB vector store.

Architecture:

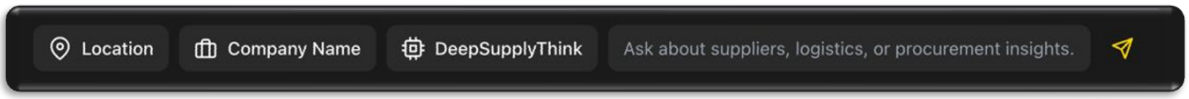

Figure: Our interface. By clicking on location or company name, we retrieve the company information with fuzzy search. If clicked on DeepSupplyThink, the user can ask broader questions, where RAG with vector search through cosine similarity will be used.

Our chatbot supports four main functionalities, which can be accessed through a GUI (FastAPI, React and TailwindCSS):

1. Default - Single-Shot RAG: With our knowledge base in place, performs semantic vector search with the embedded query. We retrieve the top-10 documents, from which the agent generates a response.
2. DeepSupplyThink - Multi-Agent system: Based on our RAG system, we refine the answers through an enhancer, who augments the user query given our prespecified categories of interest, an orchestrator that routes the enhanced query through appropriate agents (in this demo, only one, but we attempted specialized agents for

different goals, some semantic based, others SQL based). Then, a retriever queries the vector database. A sinthesizer combines the retrieved evidence of all agents (in this demo, one), and generates a coherent, context-aware answer tailored to a supply chain director. Finally, an evaluator verifies the final response for completeness, factual grounding, and alignment with the original query. If it flags the response as not valid, it sends the context to the initial agent to start again, giving it feedback.

3. Location: Based on our SQL database, where we store companies by location, the user can input a location and aggregate data like the number of companies in that area is returned.
4. Company deep-dive: When prompted about a particular company, we find the relevant documents through fuzzy matching on the company name. The company name is extracted from the query using an LLM call. With the company-related documents (.json), we craft a data-driven response to the query. If a company is not in the database it will let the user know (next step: augment the database)

Evaluation:

In order to evaluate the retrieval performance of our model, we tailor 2 datasets, where We sample 100 companies and ask the LLM to generate a question concerning a web page such that the company is the answer. The differences are:
Easy: The questions are always concerning the homepage.
http://movephoenix.com: *What companies offer comprehensive relocation services and expert real estate guidance in Arizona?*
Hard: The questions are taken from a random subpage from our cleaned dataset.
https://gravitydiagnostics.com/gravity-diagnostics-expands-testing-services-to-include-xylazine-detection-a-crucial-advancement-in-ensuring-public-safety/
"What companies are recognized for their commitment to public health through advanced toxicology testing, including newly expanded services for Xylazine detection?"
Our RAG system obtains the following recalls@10: **Easy:** 0.36 **Hard:** 0.26.

Interface:

The frontend was done with FastAPI and the backend with React and TailwingCSS.

Biggest challenges: One of the biggest challenges was to process the text to compute the embeddings. Because of the rate limit of 60 calls per minute, we could not process the whole set of pages. We arrived at a compromise and just embedded the home page of every company. Most likely, the performance could be improved if the whole set of pages was embedded.