

Appian AI Application Challenge

Github Link : https://github.com/Sudharshana426/Document_AI

Team Healix

AI-Powered Financial Document

Classification & Insights

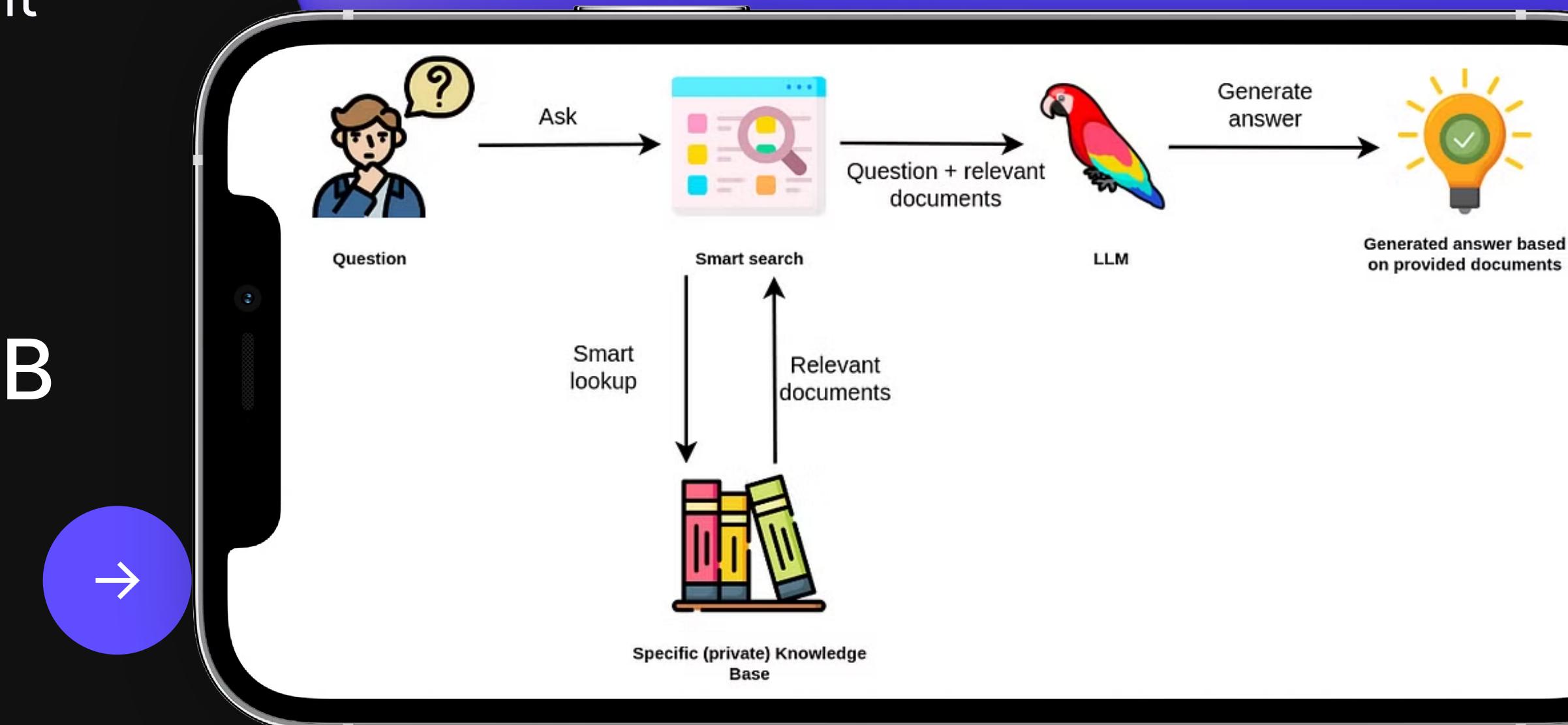
Project By:

Sudharshana B

Vishal V D

Karthik V

Transforming Unstructured Financial
Documents into Actionable Intelligence



The problem —

Appian Credit Union processes thousands of unstructured financial documents daily, including:

- Applications for bank accounts (credit card, savings account)
- Identity documents (driver's license, ID cards, passports)
- Supporting financial documents (income statements, pay stubs, tax returns)
- Receipts
- This manual process is:
 - Time-consuming and error-prone
 - Repetitive and inefficient

Every year, financial institutions like Appian Credit Union spend thousands of hours manually handling unstructured documents. This not only costs millions of dollars but also creates inefficiencies, delays, and frustration for customers and employees alike.

Solution

- **Our AI-Powered System:**

- Automatically classifies and processes unstructured financial documents.
- Links documents to individuals using metadata like names, IDs, and email addresses.
- Generates business intelligence insights for decision-making.

- **Key Features:**

- Hierarchical Classification: Person → Document Type
- Intelligent Document Association using NLP and embeddings
- Actionable insights via LLama Model

Output

Query ⇒ What are some items that Pirkko Koskitalo is likely to buy next? What incentives can I put in place to ensure he orders more?

Response: Based on the provided context, Pirkko Koskitalo has placed multiple purchase orders in the past. Analyzing these orders can help identify potential future purchases and suitable incentives.

Firstly, let's examine the repurchasing patterns of Pirkko Koskitalo. We can observe that two specific products have been ordered multiple times:

1. **Chai (Product ID: 1):** Ordered in 2017 (Order ID: 10526) with a quantity of 8, and in 2018 (Order ID: 11025) with a quantity of 10. This indicates a consistent demand for this product.
2. **Konbu (Product ID: 13):** Also ordered in 2017 (Order ID: 10526) with a quantity of 10, and in 2018 (Order ID: 11025) with a quantity of 20. This shows an increasing demand for this product.

Output

Query ⇒ What are some items that Pirkko Koskitalo is likely to buy next? What incentives can I put in place to ensure he orders more?

Considering these repurchasing patterns, Pirkko Koskitalo is likely to order these products again in the future.

Additional observations:

1. Order frequency and timing: Pirkko Koskitalo's purchase orders are made in **intervals of 6-12 months**. This might indicate a **regular replenishment schedule**.
2. Product category: **Many** of the products ordered **belong** to the category of **specialty foods (e.g., Queso Cabrales, Pavlova, Gustaf's Knäckebröd)**. This might be a **specific area of interest** for Pirkko Koskitalo.

Output

Query ⇒ What are some items that Pirkko Koskitalo is likely to buy next? What incentives can I put in place to ensure he orders more?

Incentives to encourage more orders:

1. Discounts for bulk purchases: **Offer discounts for larger quantities of Chai and Konbu** to incentivize Pirkko Koskitalo to increase their order sizes.
2. Loyalty program: Implement a **loyalty program that rewards repeat customers** for their purchases. This could be in the form of points, discounts, or exclusive access to new products.
3. Personalized promotions: Offer **targeted promotions** based on Pirkko Koskitalo's buying history and interests, such as **notifications about new specialty food** products or special discounts on their favorite items.
4. Regular communication: **Establish a regular communication channel** to inform Pirkko Koskitalo about new products, seasons, and other relevant information **that may trigger a purchase.**

By providing these incentives and understanding the buying behavior of Pirkko Koskitalo, **you can encourage more orders and establish a long-term relationship with this customer.**

Competitive advantages

1

What makes US unique?

- Combines document classification, person association, and business insights into one seamless solution from the AI Chatbot.
- Scalable to **handle diverse document types** and datasets.
- Interactive AI chatbot** for real-time insights and recommendations.

2

How will you be able to

- ### outperform your competitors?
- Generate trends and actionable insights (e.g., document volume, missing fields).

- Provide metrics for process optimization and customer demographics.



- ### Example Outputs:
- Average processing time for applications.
 - Trends in document submissions (e.g., receipts by month).
 - Errors in identity documents (e.g., incomplete fields).

Overwhelmed by **unstructured documents**, making it **difficult**
to extract valuable insights →

Automates the extraction and analysis of data from company
documents →

Transforms thousands of documents into **searchable, valuable**
business intelligence using **RAG** →

With a dataset of over **2,677** documents, my solution reduced
manual processing time →

The screenshot shows a digital interface for managing company documents. At the top, there's a search bar with a magnifying glass icon and the word "Search". Below the search bar, the title "Company Documents Dataset" is displayed in bold white text. Underneath the title, there are several tabs: "Data Card" (which is underlined, indicating it's the active tab), "Code (3)", "Discussion (0)", and "Sugge...". Below the tabs, there's a toolbar with various icons for filtering and sorting data. The main content area displays an "Order ID: 10248" section. This section includes "Shipping Details:" with information: Ship Name: Vins et alcools Chevalier, Ship Address: 59 rue de l'Abbaye, Ship City: Reims, Ship Region: Western Europe, Ship Postal Code: 51100, and Ship Country: France. It also includes "Customer Details:" with Customer ID: VINET and Customer Name: Vins et alcools Chevalier. Finally, it shows "Employee Details:" with Employee Name: Steven Buchanan.

Company Documents Dataset

Data Card Code (3) Discussion (0) Sugg...

Order ID: 10248

Shipping Details:

Ship Name: Vins et alcools Chevalier
Ship Address: 59 rue de l'Abbaye
Ship City: Reims
Ship Region: Western Europe
Ship Postal Code: 51100
Ship Country: France

Customer Details:

Customer ID: VINET
Customer Name: Vins et alcools Chevalier

Employee Details:

Employee Name: Steven Buchanan

For example dataset from Kaggle :

<https://www.kaggle.com/datasets/ayoubcherguelaine/company-documents-dataset>

What?

This project focuses on
automating the extraction and analysis of structured
sales dataset from over 2677 company documents
using AI-powered RAG techniques.

- Invoices
- Purchase orders
- Inventory reports
- Shipping orders

Why?

Automating the extraction and analysis of
company documents will save significant
time and resources, reducing manual effort
and errors.

- Rapid document processing
- Automated extraction
- Easy access to key insights
- Focus on high-value tasks

Project Workflow

Setup & Installation



Data Preprocessing



Text Embedding



Vector Indexing



Query Embedding and Matching



Retrieval-Augmented Generation (RAG)



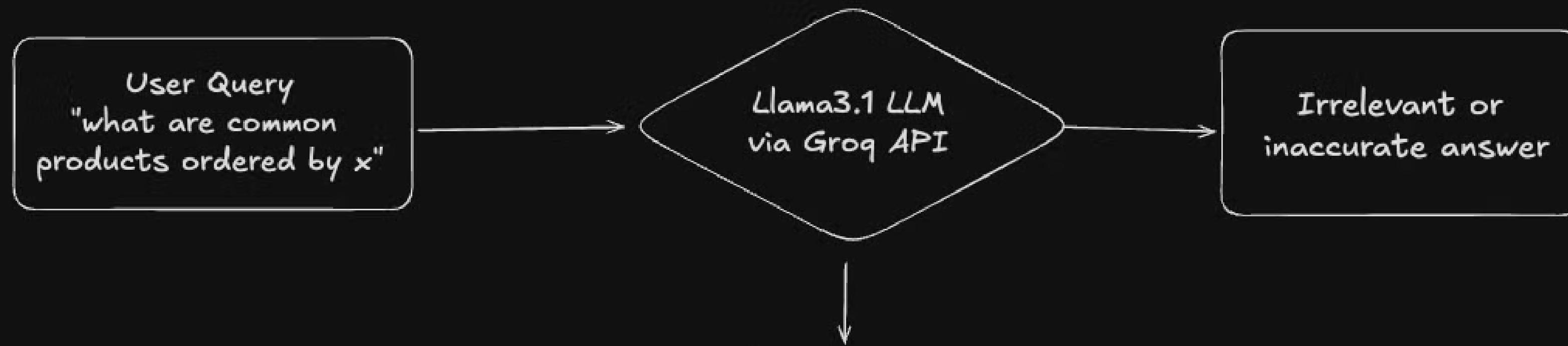
LLM Integration



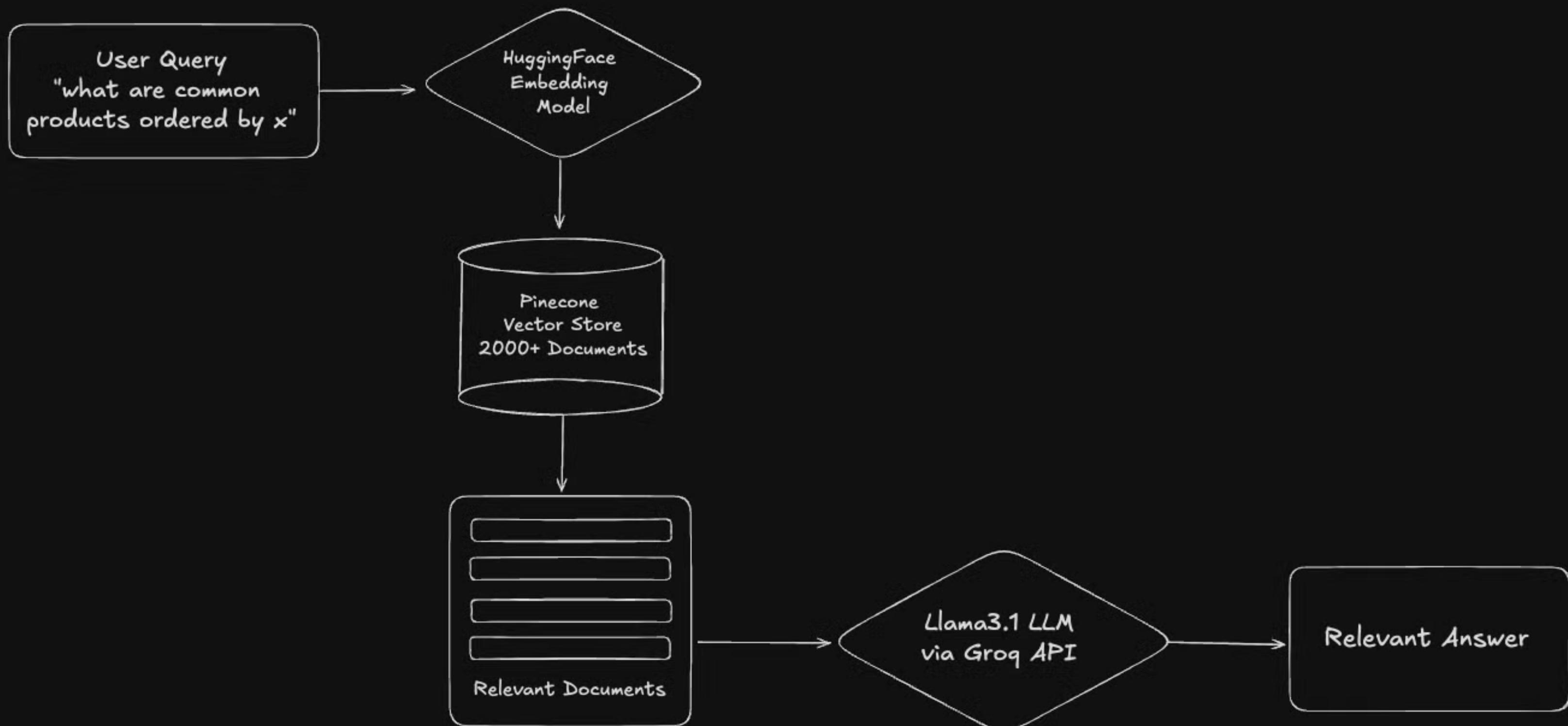
Response Generation



Without RAG



With RAG



Setup & Installation

We need to install several Python packages that will help us with various tasks like processing data, embedding conversion and connecting to APIs.

```
! pip install langchain langchain-community openai groq tiktoken pinecone-client langchain_pinecone
Requirement already satisfied: langchain in /usr/local/lib/python3.10/dist-packages (0.3.7)
Collecting langchain-community
  Downloading langchain_community-0.3.9-py3-none-any.whl.metadata (2.9 kB)
Requirement already satisfied: openai in /usr/local/lib/python3.10/dist-packages (1.54.4)
Collecting groq
  Downloading groq-0.13.0-py3-none-any.whl.metadata (13 kB)
Collecting tiktoken
  Downloading tiktoken-0.8.0-cp310-cp310-manylinux_2_17_x86_64_manylinux2014_x86_64.whl.metadata (19 kB)
Collecting pinecone-client
  Downloading pinecone_client-5.0.1-py3-none-any.whl.metadata (19 kB)
```

- langchain
- langchain-community
- openai
- Groq
- Tiktoken
- pinecone-client
- langchain_pinecone
- unstructured_inference
- sentence-transformers

Data Preprocessing

```
▶ from langchain.document_loaders import UnstructuredPDFLoader, OnlinePDFLoader
from langchain.text_splitter import RecursiveCharacterTextSplitter
from sklearn.metrics.pairwise import cosine_similarity
from langchain_pinecone import PineconeVectorStore
from langchain.embeddings import OpenAIEmbeddings
from langchain_community.embeddings import HuggingFaceEmbeddings
from google.colab import userdata
from langchain.schema import Document
from sentence_transformers import SentenceTransformer
from pinecone import Pinecone
from openai import OpenAI
import numpy as np
import tiktoken
import os
from groq import Groq
```

After extracting text, it is often necessary to preprocess the data by removing unwanted characters or handling missing values. Here, we load the CSV containing document labels and clean it by removing rows with missing text.

- UnstructuredPDFLoader
- OnlinePDFLoader
- WebBaseLoader
- YoutubeLoader
- DirectoryLoader
- TextLoader
- PyPDFLoader

Text Embedding

```
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")  
  
ipython-input-3-3a8dd36e4865>:1: LangChainDeprecationWarning: The class `HuggingFaceEmbeddings` was depr  
embeddings = HuggingFaceEmbeddings(model_name="sentence-transformers/all-mpnet-base-v2")  
/usr/local/lib/python3.10/dist-packages/huggingface_hub/utils/_auth.py:94: UserWarning:  
The secret `HF_TOKEN` does not exist in your Colab secrets.  
To authenticate with the Hugging Face Hub, create a token in your settings tab (https://huggingface.co/settings/secrets/tokens). You will be able to reuse this secret in all of your notebooks.  
Please note that authentication is recommended but still optional to access public models or datasets.  
warnings.warn(  
modules.json: 100% [00:00<00:00, 11.5kB/s]  
config_sentence_transformers.json: 100% [00:00<00:00, 3.61kB/s]  
README.md: 100% [00:00<00:00, 297kB/s]  
sentence_bert_config.json: 100% [00:00<00:00, 1.74kB/s]  
config.json: 100% [00:00<00:00, 38.5kB/s]  
model.safetensors: 100% [00:02<00:00, 188MB/s]  
tokenizer_config.json: 100% [00:00<00:00, 20.4kB/s]  
vocab.txt: 100% [00:00<00:00, 3.92MB/s]  
tokenizer.json: 100% [00:00<00:00, 8.98MB/s]  
special_tokens_map.json: 100% [00:00<00:00, 9.49kB/s]  
1_Pooling/config.json: 100% [00:00<00:00, 11.3kB/s]
```

The Hugging Face Embeddings class acts as a wrapper around Hugging Face models, allowing users to easily generate embeddings for texts. It requires the sentence_transformers package to be installed.

- Hugging Face Embeddings
- Sentence Transformers library
- langchain-huggingface

Developer quickstart

Learn how to get started with Pinecone and start building your first application.

[View guide](#)

```
1 # Import the Pinecone library
2 from pinecone import Pinecone
3
4 # Initialize a Pinecone client with your API key
5 pc = Pinecone(api_key="YOUR_API_KEY")
```

```
[ ] pinecone_api_key = userdata.get("PINECONE_API_KEY")
os.environ['PINECONE_API_KEY'] = pinecone_api_key

index_name = "codebase-rag"

namespace = "company-documents"

vectorstore = PineconeVectorStore(index_name=index_name, embedding=embeddings)
```

Vector Indexing

Indexing document embeddings for efficient retrieval. Configure Pinecone for scalable vector storage for storing document embeddings with metadata for querying.

- Pinecone Vector Store
- pinecone-client
- langchain_pinecone

codebase-rag •

Host: <https://codebase-rag-42whbt9.svc.aped-4627-b74a.pinecone.io> 

Cloud: AWS • **Region:** us-east-1 • **Capacity mode:** Serverless • **Dimension:** 768

...

[Connect](#)

Query Embedding and Matching

```
▶ def get_huggingface_embeddings(text, model_name="sentence-transformers/all-mnlp-base-v2"):
    model = SentenceTransformer(model_name)
    return model.encode(text)

def cosine_similarity_between_sentences(sentence1, sentence2):
    embedding1 = np.array(get_huggingface_embeddings(sentence1))
    embedding2 = np.array(get_huggingface_embeddings(sentence2))

    embedding1 = embedding1.reshape(1, -1)
    embedding2 = embedding2.reshape(1, -1)

    print("Embedding for Sentence 1:", embedding1)
    print("\nEmbedding for Sentence 2:", embedding2)

    similarity = cosine_similarity(embedding1, embedding2)
    return similarity[0][0]

sentence1 = "I like walking to the park"
sentence2 = "I like running to the office"

similarity = cosine_similarity_between_sentences(sentence1, sentence2)
print(f"\n\nCosine similarity between '{sentence1}' and '{sentence2}': {similarity:.4f}")
```

Efficiently retrieve contextually relevant documents.

Generate embeddings for user queries.

Perform cosine similarity searches to find top document matches.

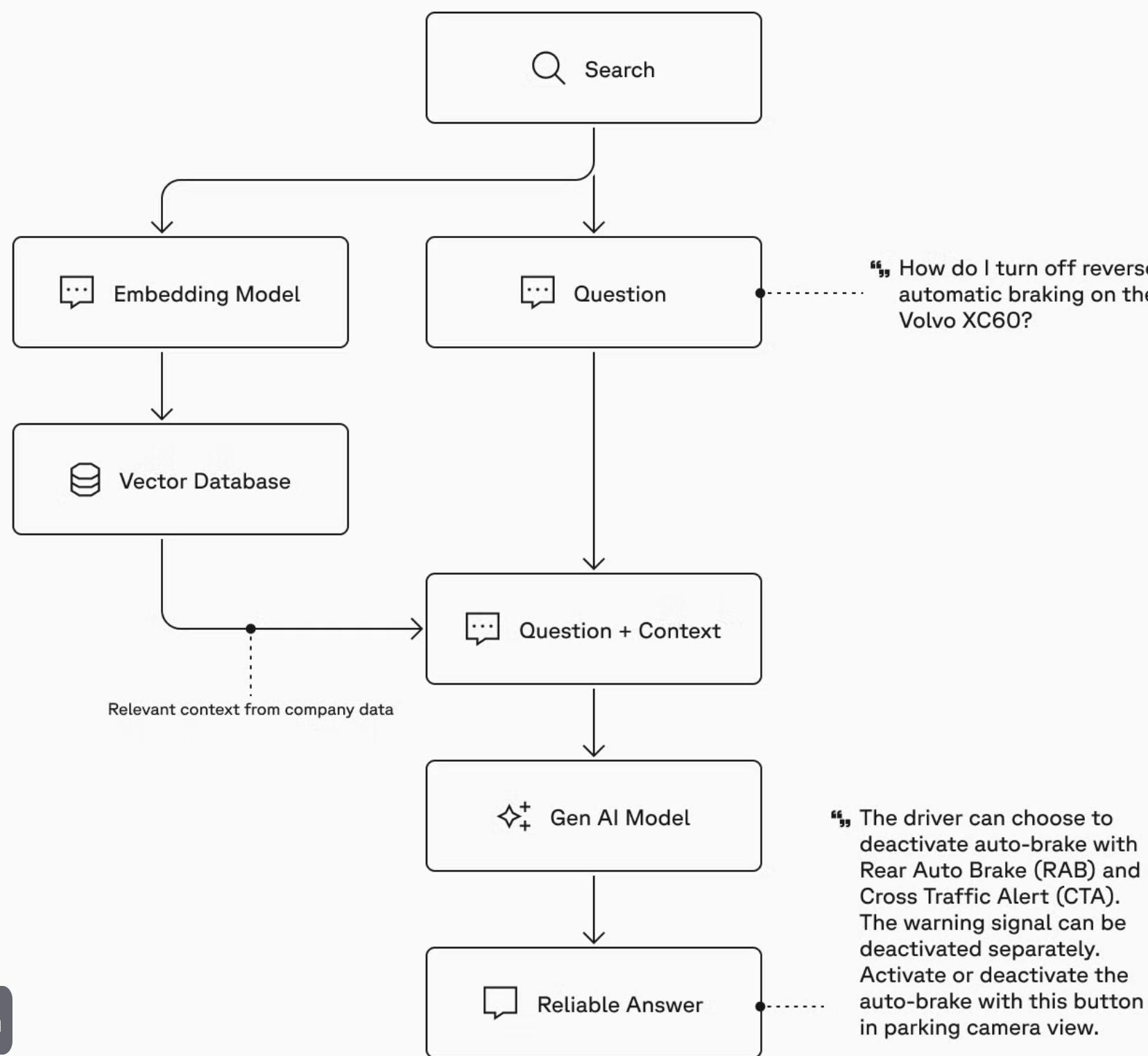
Integrate queries with Pinecone for real-time retrieval.

cosine_similarity (NumPy, Scikit-learn)

Pinecone Query APIs

```
[ ] Vectorstore_from_documents = PineconeVectorStore.from_documents(
    document_data,
    embeddings,
    index_name=index_name,
    namespace=namespace
)
```

Retrieval-Augmented Generation (RAG)



Query the vector store for top-matching documents based on user input.

Retrieve relevant context snippets to enhance the query.

Combine document context with user questions to create an augmented query.

- Pinecone Query APIs
- Custom RAG Pipelines
- Query-Augmentation Functions

groq | Meta

Supercharging
Fast AI Inference for
Llama 3.1

New Largest Openly Available
Foundation Model 405B

Running on GroqCloud™

system_prompt = f"""You are an expert at understanding and analyzing company data

Answer any questions I have, based on the data provided. Always consider all of the
"""

```
llm_response = groq_client.chat.completions.create(  
    model="llama-3.1-70b-versatile",  
    messages=[  
        {"role": "system", "content": system_prompt},  
        {"role": "user", "content": augmented_query}  
    ]  
)
```

Try Pitch

LLM Integration

Leveraging advanced language models for intelligent responses.

- Use **Groq's LLM API ⇒ Llama-3.1-70b-versatile** to process the augmented query.
- Integrate system prompts that instruct the model to analyze specific company documents (e.g., invoices, purchase orders).
- The LLM generates responses based on the retrieved context and user query.

System Prompt Creation

Query Processing

Response Generation

```
groq_api_key = userdata.get("GROQ_API_KEY")  
os.environ['GROQ_API_KEY'] = groq_api_key
```

```
groq_client = Groq(api_key=os.getenv('GROQ_API_KEY'))
```

```
print(response)
```

Based on the provided context, Pirkko Koskitalo has placed multiple purchase orders in the past. Analyzing these orders can help identify potential future purchases and suitable incentives.

Firstly, let's examine the repurchasing patterns of Pirkko Koskitalo. We can observe that two specific products have been ordered multiple times:

1. **Chai (Product ID: 1)**: Ordered in 2017 (Order ID: 10526) with a quantity of 8, and in 2018 (Order ID: 11025) with a quantity of 10. This indicates a consistent demand for this product.
2. **Konbu (Product ID: 13)**: Also ordered in 2017 (Order ID: 10526) with a quantity of 10, and in 2018 (Order ID: 11025) with a quantity of 20. This shows an increasing demand for this product.

Considering these repurchasing patterns, Pirkko Koskitalo is likely to order these products again in the future.

Additional observations:

1. **Order frequency and timing**: Pirkko Koskitalo's purchase orders are made in intervals of 6-12 months. This might indicate a regular replenishment schedule.
2. **Product category**: Many of the products ordered belong to the category of specialty foods (e.g., Queso Cabrales, Pavlova, Gustaf's Knäckebröd). This might be a specific area of interest for Pirkko Koskitalo.

Incentives to encourage more orders:

1. **Discounts for bulk purchases**: Offer discounts for larger quantities of Chai and Konbu to incentivize Pirkko Koskitalo to increase their order sizes.
2. **Loyalty program**: Implement a loyalty program that rewards repeat customers for their purchases. This could be in the form of points, discounts, or exclusive access to new products.
3. **Personalized promotions**: Offer targeted promotions based on Pirkko Koskitalo's buying history and interests, such as notifications about new specialty food products or special discounts on their favorite items.
4. **Regular communication**: Establish a regular communication channel to inform Pirkko Koskitalo about new products, seasons, and other relevant information that may trigger a purchase.

By providing these incentives and understanding the buying behavior of Pirkko Koskitalo, you can encourage more orders and establish a long-term relationship with this customer.

Response Generation

- Delivering actionable insights to end-users.
- Parse LLM responses into concise, human-readable outputs.
- Highlight trends, actionable insights, and recommendations based on query context.

Query ⇒ What are some trends with André Fonseca purchase orders?

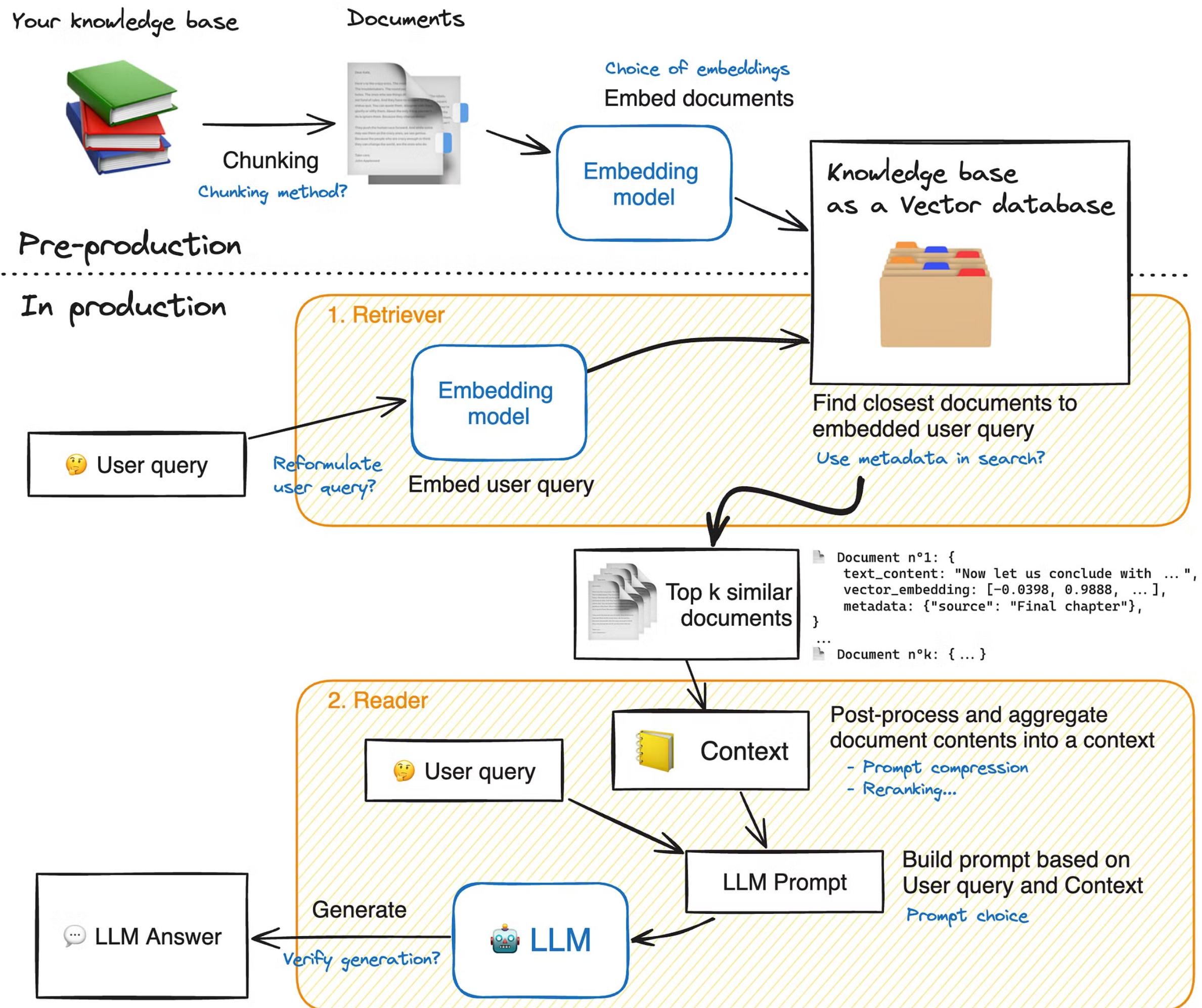
There's only **one purchase order** in the provided data for André Fonseca, which is **Order ID 10423**, placed on **2017-01-23**.

From this single purchase order, some initial observations can be made:

- 1. Product selection:** André Fonseca ordered Gorgonzola Telino (Product ID: 31) and Raclette Courdavault (Product ID: 59). This suggests that he might be **interested in high-end or specialty cheese products**.
- 2. Order size:** The order contains 14 units of Gorgonzola Telino and 20 units of Raclette Courdavault. Since this is only one order, it's difficult to establish a pattern of order size, but this could be **indicative of a moderate or small business demand**.
- 3. Unit prices:** The unit prices for the ordered products are 10 and 44 respectively, which could **imply that André Fonseca might be a customer who values high-quality products, even at a higher cost**.
- 4. Occasional buying pattern:** This is only one order, but it's worth noting that if André Fonseca orders sporadically or only occasionally, **his buying patterns might not follow a consistent trend**.

Keep in mind that these observations are based on a single purchase order. Further data on André Fonseca's purchase orders would be required to confirm these trends or identify more substantial patterns.

W O R K F L O W



Feasibility & Scalability

Modular Design:

Adapts to varied datasets with minimal changes.
Scalable for large document volumes using vector storage.

Efficient Performance:

Real-time retrieval with Pinecone for embeddings.
Supports multiple formats (PDFs, images, text files).

Summary

Our Solution focuses on creating an advanced **AI-Driven Document Chatbot** that leverages **Natural Language Processing (NLP)** and **embedding-based models** for analyzing and retrieving insights from a dataset containing over **2,000 + company documents**. Using tools like

- Pinecone for vector similarity**
- HuggingFace embeddings**
- LangChain for contextual query answering**

the system ensures efficient information extraction and enhanced decision-making capabilities.

Appian AI Application Challenge

Team Healix

Project By:

Sudharshana B
Vishal V D
Karthik V



Thank
You