A Custom Knowledge Base Chatbot Using Retrieval Augmented Generation (RAG)

Orange Oracle: Christian Ortmann, Dhawal Ajay Gajwe, Sreeharsha Nalluri, Venkata Mahesh Kundurthi

Abstract

This Project describes the construction and execution of a Retrieval-Augmented Generation (RAG)-based domain-adaptable chatbot capable of contextually aware factual reasoning on user-supplied knowledge bases. Like many traditional large language models, LLMs tend to hallucinate outputs when queried about subject matters they have not been exposed to during the training processes. On the other hand, RAG boosts reliability of responses by integrating document retrieval with LLM generation through a querying process on a knowledge vector store. The system proposed here has implemented Langchain and OpenAI's embedding model with a Supabase vector store for enhanced semantic answer-finding capabilities. In the backend, processes performed on documents uploaded as fed through chunking into embeddings, which are stored to and retrieved from a cosine similarity guided generative process. The user-facing frontend is built on Next.js and styled with Tailwind CSS to enable interactivity when uploading documents and querying the AI across conversations. While we encountered open-source LLMs integration challenges via CyVerse due to infrastructure and tools, final deployment on commercial APIs exposed strong performant practicality. This project demonstrates the critical role of retrieval mechanisms in reducing hallucinated responses by AI models. It also proves that scalable and customizable AI-human interactions are achievable when tailored to specific user-defined domains or reasoning tasks.

Introduction

Chatbots are extremely useful tools that leverage AI to create near human responses to natural language prompts. When it comes to generalized models that are trained on a vast variety of information from across domains, these chatbots can hallucinate responses that sound real yet are completely false when prompted to answer a question about a hyper specific subject area. Retrieval-Augmented Generation (RAG) improves this by generating responses that are improved by information that the model is able to retrieve from a given source (Lewis et al., 2021). RAGs are able to generate more refined responses because they have information not only from pre-training, but information that is retrieved from documentation sources like Wikipedia or Britannica or a Vector Database. This project focuses on the development of a Custom Knowledge Base Chatbot that will leverage the Retrieval-Augmented Generation (RAG) framework to enhance human-AI interactions through domain-specific, context-aware responses. This chatbot will adopt the framework of a RAG, but the knowledge base, which the model retrieves from, will be a custom repository for the user to train the model on their problem specific information, allowing the user to upload or remove documents, allowing flexible adaptation to various domains. By integrating document retrieval with generative AI, the chatbot can access and synthesize information dynamically from the custom knowledge base, which enables it to address user queries with higher relevance and accuracy. This approach has applications in domains where a custom knowledge base may be confidential, such as AI therapy, and can generate responses based purely on the information fed into the model, in this case a therapist's notes on a particular patient. Through this project, we aim to deepen our understanding of the RAG pipeline, analyze how intelligent agents utilize retrieved knowledge, and showcase a real-world use case that demonstrates the practical value of RAG-powered systems.

Related Work

Past research suggests that document supported outputs from RAG frameworks display a compelling case for real world implementation. One of the first publications in the literature was Lewis et al. (2021), which presented the concept of the RAG framework with a non-parametric memory dense vector index of information from Wikipedia that complemented the parametric memory of a traditional Large Language Model (LLM). Here, the results showed that the RAG framework tended to generate more factual and specific language responses than purely parametric models such as BART, paving the way for further implementation with lower rates of "hallucinated" responses.

Given three years to refine the RAG framework, similar work was published for fact checking to battle disinformation in 2024, including Khaliq et al. (2024) and Russo et al. (2024). Khaliq et al. focused on the implementation of RAG and two new frameworks: CoRAG and Tree of RAG, methods that break queries down into steps or a "chain" as in CoRAG, or breaking the queries off into branches of evidence elimination to enhance response accuracy as in Tree of RAG. This paper found that an increase of 0.06-0.14 F1 score was possible when implementing these frameworks for fact checking. Russo et al. found that LLM based retrievers outperform other retrievers, such as dense or hybrid, in the case of homogenous knowledge bases, but had some degree of difficulties with heterogeneous knowledge bases. They suggested that the best path forward may be with a RAG that utilizes a hybrid retriever which can balance computational cost and accuracy. These three papers suggest that there is a very large upside to utilizing the RAG framework, especially when it comes to reducing hallucination in natural language responses.

The papers were not without limitations however, as Khaliq et al. utilized a closed source LLM that may not be widely accessible, while Russo et al. only implemented their framework in the English language, and Lewis et al. pulled their knowledge base from Wikipedia, an information repository that can be modified by anyone anywhere in the world and can contain factually wrong information. Thus it is important to always review these papers with an eye for potential limitations, though in these papers, they are small and do not draw from the overall impact of the implications of the findings.

<u>Methodology</u>

Using the Retrieval-Augmented Generation paradigm, this project creates a domain-adaptable chatbot. The framework incorporates LangChain, large models from OpenAI, and a language interface over a custom knowledge base powered by a Supabase-hosted vector store to allow fluent interactions through natural language. The pipeline includes three interwoven parts: the knowledge base data acquisition and processing pipeline, LLM model in the AI system, and the framework hosting interface which also manages the software engineering functionalities. In this chapter, the methods are articulated with precision, describing the technical details of the implementation as well as design decisions at the system level.

Data Pipeline

The data pipeline, shown in Figure 1, processes user documents into chunks that are organized for efficient retrieval during queries. The data pipeline is capable of supporting regular updates to the knowledge base, such as enabling users to modify or remove domain-specific documents.

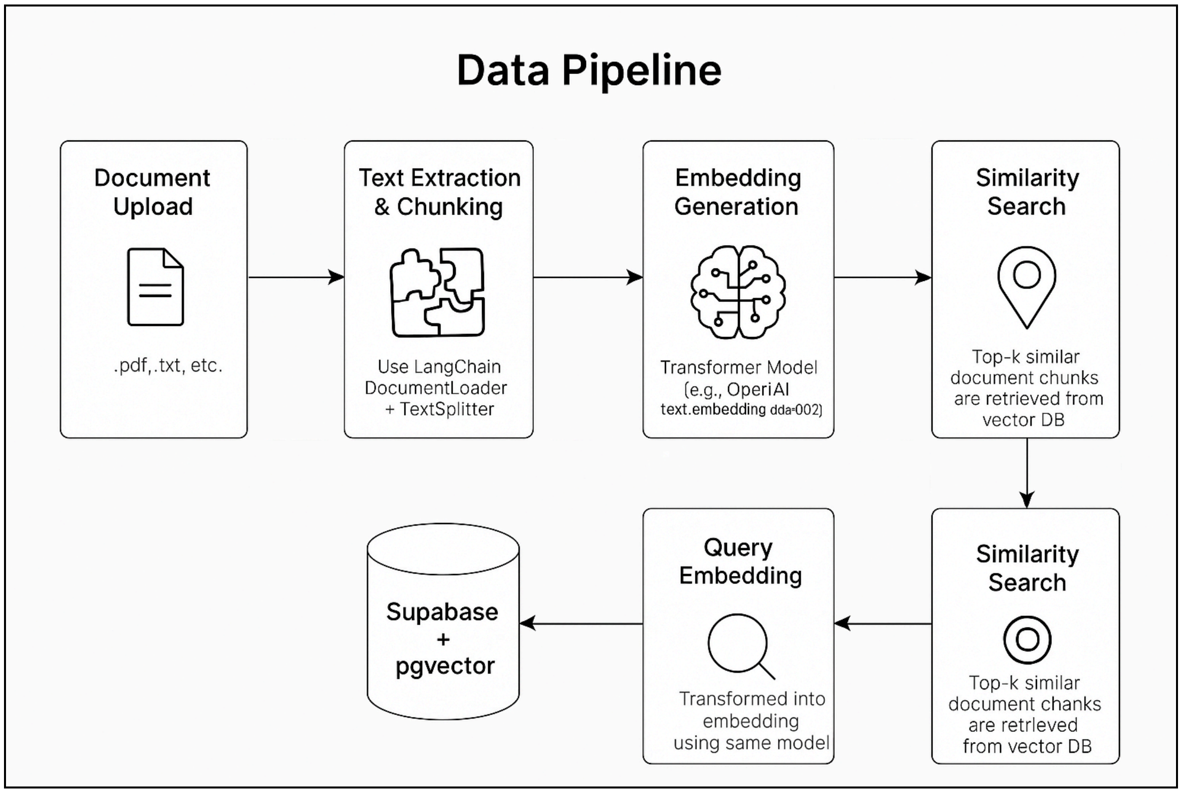

Data Pipeline
Document Upload
.pdf,.txt, etc.
Text Extraction & Chunking
Use LangChain DocumentLoader + TextSplitter
Embedding Generation
Transformer Model (e.g., OperiAI text.embedding dda-002)
Similarity Search
Top-k similar document chunks are retrieved from vector DB
Similarity Search
Top-k similar document chanks are retrleved from vector DB
Query Embedding
Transformed into embedding using same model
Supabase + pgvector

Figure 1: Breakdown of data pipeline

Users can submit knowledge base text through the web-based interface (Currently we only support text input). After the knowledge base is uploaded, the backend server is provided with the text. The extracted text is divided into smaller overlapping chunks using LangChain's Recursive Character TextSplitter. Typically, chunks are capped at 500 to 1000 characters with an overlap of 100 to 200 characters, depending on configuration. Improved retrieval accuracy and decreased context loss because of the overlapping nature between chunks improves semantics retention, ensuring contextual inclusion during tokenization.

A transformer-based sentence embedding model encodes each chunk into its corresponding dense vector. The system is compatible with OpenAI's text-embedding-ada-002 (using API). These models capture semantic meaning and are very important for similarity search. The chunk embeddings are stored in a PostgreSQL Supabase database that is hosted and managed by Supabase which has pgvector extension. Every vector entry is stored with identifying metadata: chunk index, document name, and timestamp. This enables fast document retrieval while providing clear traceability to source documents. Supabase also includes file storage, REST APIs, authentication, and other features that simplify backend management. The user-submitted question to the chatbot is embedded as well. A cosine similarity search is then done on the stored vectors to return the top-k relevant document chunks, which serve as the context for generating the final answer.

Model Architecture

The framework uses LangChain to incorporate document retrieval, answer formulation into a unified system. Its modular structure supports additional tools or agents, expanding its adaptability to include customer service AI or tutoring software. This project implements a traditional RAG (retrieval-augmented generation) pipeline.The retriever looks through the data to find the most relevant document, and the generator uses that content to write a response. Unlike models that rely only on what they've been trained on, this method pulls directly from real sources, making the answers more accurate and easier to fact-check. An agent is implemented in LangChain for controlling the logic that generates an answer. It creates custom prompts, executes the language model, and adds relevant tools like vector search or document summarizers. Agents are defined with toolkits and prompting templates for precise command oversight conducive to intelligent management of tool engagement.

The prompt templates facilitate retrieval of document snippets which are formatted in a way comprehensible to the LLM so they can be prepended to the prompt that is already provided by the user. This allows the model to reason externally and not just from knowledge found within the training data the model was built upon. For multi-turn conversations, LangChain's Conversation Buffer Memory retains and manages prior exchanges as memory. This history is added to the following prompts, making it possible for the model to have a sense of continuity and coherence throughout user sessions. The given architecture is incorporated with a Next.js and Tailwind CSS based frontend. The users can upload text documents, start a chat, and see source linked responses. The responsive and easy to use interface allows backend components to be accessed via API calls, making it easier to manage.

Deployment and Hosting
To make the Custom Knowledge Base Chatbot accessible to a broader audience, the application has been deployed live using Vercel, a platform optimized for frontend frameworks like Next.js. Hosting the application on Vercel enables seamless access via a shareable URL, removing the need for local setup and making it instantly available to any user with internet access.

The frontend, built with Next.js and Tailwind CSS, is tightly integrated with backend APIs and vector search functionality. Upon deploying to Vercel, environment variables such as API keys, Supabase credentials, and embedding model configurations are securely managed via Vercel's project settings. This ensures a secure, production-ready deployment with minimal latency.

The hosted app allows users to:

- Upload their own documents to build a personalized knowledge base
- Engage with the chatbot in a dynamic, conversational interface
- View source-linked responses powered by real-time vector retrieval

Vercel's global content delivery network (CDN) ensures low-latency access and fast load times, regardless of user location. Additionally, automatic HTTPS, CI/CD integration, and custom domain mapping (if required) provide a robust and scalable hosting environment for the RAG-powered chatbot system.

This live deployment is a crucial step in transitioning the chatbot from a development environment to a usable tool that can be tested, demonstrated, or even adopted by others in real-world scenarios.

Challenges

During the development of the Custom Knowledge Base Chatbot, one of the main challenges was the attempted integration of an open-source language model hosted on the CyVerse platform. The motivation behind this was to explore alternatives to commercial APIs, aiming for greater flexibility and potential cost savings. However, the process revealed several complexities.

CyVerse, while powerful, required a solid understanding of its infrastructure, including setting up virtual machines, managing containers, and handling authentication and secure data transfer. There is no easy access to the model via an API. A new layer for handling REST calls must be added to handle responses. There is very limited documentation for accessing CyVerse API for TypeScript/JavaScript programming language. We were able to test the functionality through a sample python code but were unable to progress for our tech stack.

Due to these technical and operational hurdles, as well as time constraints, the integration could not be completed within the project timeline. As a result, the chatbot continued to rely on commercial APIs (like OpenAI's GPT models) for its core functionality. This experience highlighted the practical challenges of adopting open-source AI infrastructure, especially when compared to the relative ease and support provided by managed commercial services.

Ethical Considerations

Ethically speaking, a RAG framework can expose private information if its knowledge base is confidential. Additionally, the model will be biased towards its knowledge base, especially if it contains limited information or one sided facts. It is therefore paramount to provide a strong factual knowledge base that contains public knowledge or information that has been approved to be shared with the user.

It is also important to consider the carbon emissions generated from computationally dense models such as LLMs and to carefully consider the amount of queries required to solve a problem. Careful thought into the complexity of a query or clearly defining the task will lead to less GPU usage, helping to lower carbon emissions and promote a more environmentally friendly future.

Future Work

For future development, further exploration of open-source model integration is recommended, possibly by collaborating with the CyVerse community or seeking alternative platforms with better support. Ability to define custom database schema, the current database schema is basic for more complex problems, having a well defined database schema according to the problem statement should improve the retrieval. Streamlining deployment pipelines and improving documentation would make such integrations more feasible. Additionally, ongoing improvements to the chatbot's retrieval and generation capabilities, such as adopting advanced RAG architectures, hybrid retrieval mechanisms, and multi-modal knowledge integration, remain important. Continued attention to scalability, ethical considerations, and user experience will help ensure the chatbot remains a reliable and adaptable tool for accessing specialized information.

Conclusion

This project successfully developed and evaluated a Custom Knowledge Base Chatbot using the RAG framework. The system demonstrated strong performance in delivering domain-specific, context-aware responses by retrieving and synthesizing information from user-provided knowledge bases. While the chatbot met its primary objectives using commercial APIs, the attempt to integrate an open-source model from CyVerse revealed significant technical and operational challenges, including complex setup, limited documentation, and the need for specialized infrastructure knowledge. Ultimately, the project continued with commercial solutions to ensure reliability and timely delivery, but the experience underscored the importance of usability and support when selecting AI platforms.

References

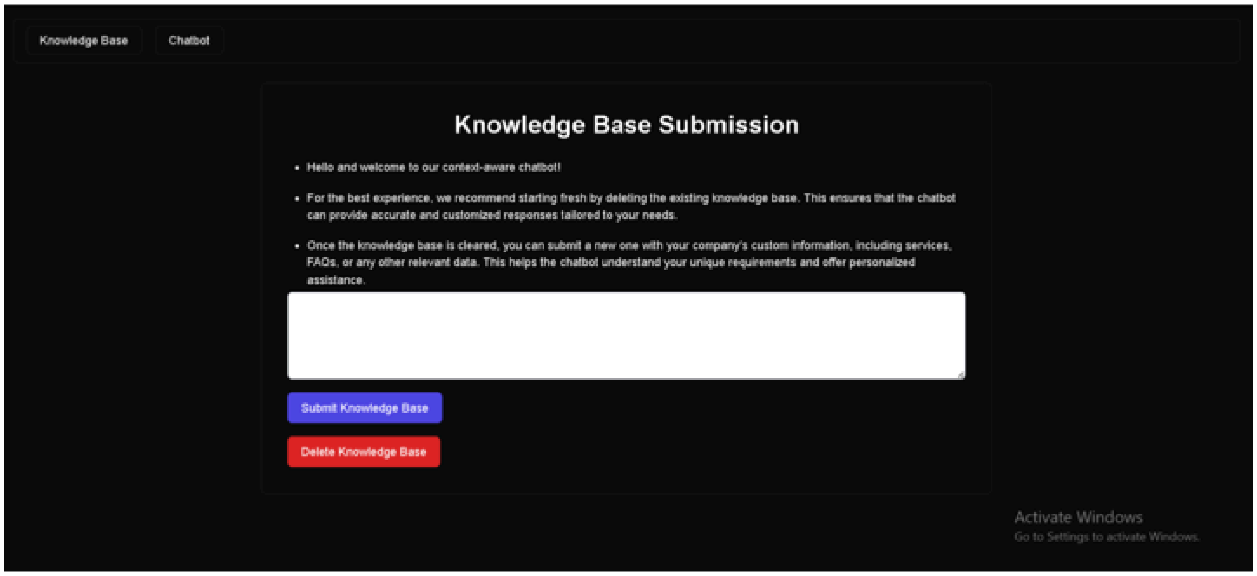
Khaliq, M. A., Chang, P., Ma, M., Pflugfelder, B., & Miletić, F. (2024). *RAGAR, Your Falsehood Radar: RAG-Augmented Reasoning for Political Fact-Checking using Multimodal Large Language Models* (No. arXiv:2404.12065). arXiv. https://doi.org/10.48550/arXiv.2404.12065

Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W., Rocktäschel, T., Riedel, S., & Kiela, D. (2021). *Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks* (No. arXiv:2005.11401). arXiv. https://doi.org/10.48550/arXiv.2005.11401

Russo, D., Menini, S., Staiano, J., & Guerini, M. (2024). *Face the Facts! Evaluating RAG-based Fact-checking Pipelines in Realistic Settings* (No. arXiv:2412.15189). arXiv. https://doi.org/10.48550/arXiv.2412.15189

Appendix A: System UI and Knowledge Base Backend

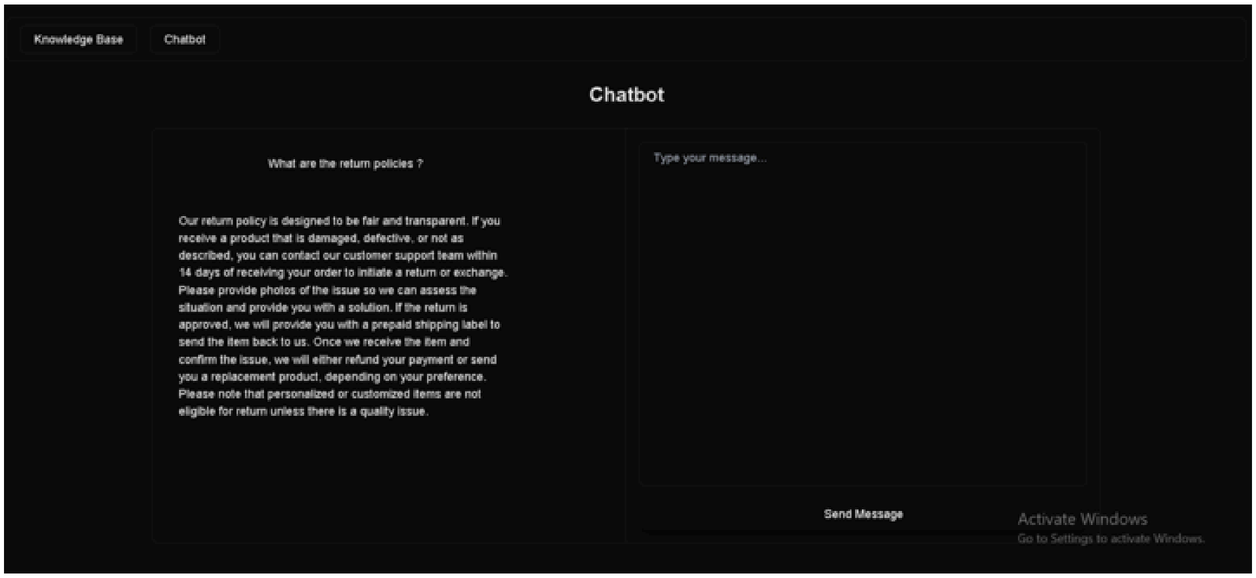
A.1: Live version of chatbot available at: https://custom-knowledge-base-rag-chatbot.vercel.app/



Knowledge Base Submission

- Hello and welcome to our context-aware chatbot!
- For the best experience, we recommend starting fresh by deleting the existing knowledge base. This ensures that the chatbot can provide accurate and customized responses tailored to your needs.
- Once the knowledge base is cleared, you can submit a new one with your company's custom information, including services, FAQs, or any other relevant data. This helps the chatbot understand your unique requirements and offer personalized assistance.

A.2: UI of knowledge base submission form, from here the knowledge base is sent to the supabase vector database where the model will pull its information from. Here, a knowledge base of a fictional store called "TheWallCo" that provides custom printing is entered and submitted. (The knowledge base is created using ChatGPT which has all the services/products TheWallCo provides).

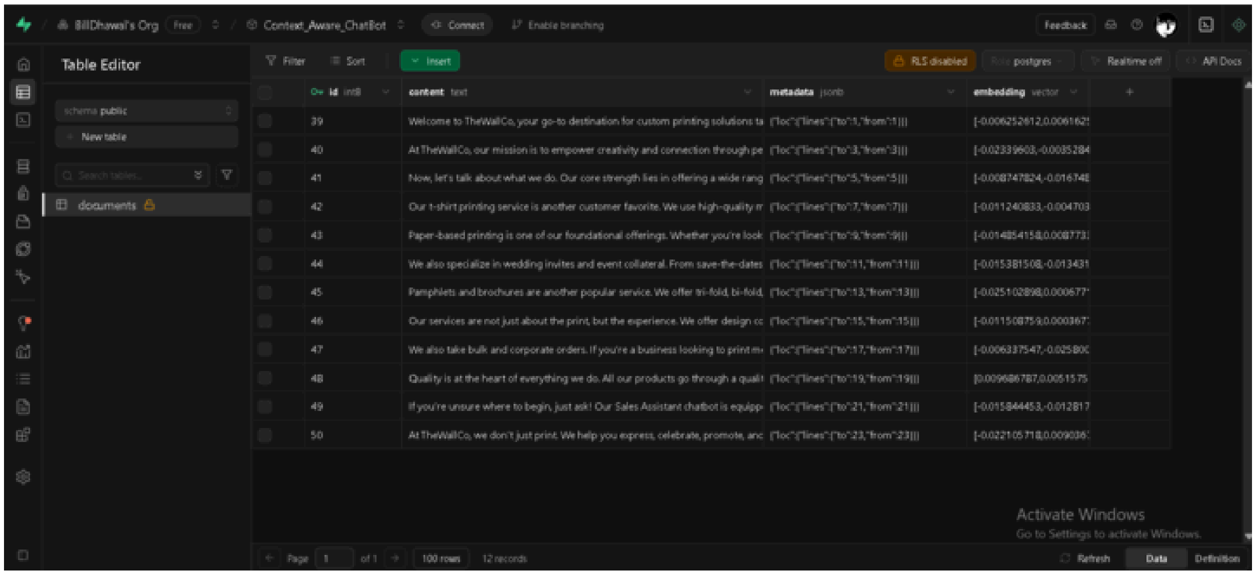


Chatbot

What are the return policies ?

Our return policy is designed to be fair and transparent. If you receive a product that is damaged, defective, or not as described, you can contact our customer support team within 14 days of receiving your order to initiate a return or exchange. Please provide photos of the issue so we can assess the situation and provide you with a solution. If the return is approved, we will provide you with a prepaid shipping label to send the item back to us. Once we receive the item and confirm the issue, we will either refund your payment or send you a replacement product, depending on your preference. Please note that personalized or customized items are not eligible for return unless there is a quality issue.

A.3: UI of chatbot. Here queries can be submitted and the RAG based model will provide factual responses based on the submitted knowledge base. Here, a query regarding the return policy of the fictional store called "TheWallCo" is entered and the return policy (based on the knowledge base entered earlier) is returned in a natural language format.



id int8	content text	metadata jsonb	embedding vector
39	Welcome to TheWallCo, your go-to destination for custom printing solutions ta	{"loc":{"lines":{"to":1,"from":1}}}	[-0.006252612,0.0061625
40	At TheWallCo, our mission is to empower creativity and connection through pe	{"loc":{"lines":{"to":3,"from":3}}}	[-0.02339603,-0.0035284
41	Now, let's talk about what we do. Our core strength lies in offering a wide rang	{"loc":{"lines":{"to":5,"from":5}}}	[-0.008747824,-0.016748
42	Our t-shirt printing service is another customer favorite. We use high-quality m	{"loc":{"lines":{"to":7,"from":7}}}	[-0.011240833,-0.004703
43	Paper-based printing is one of our foundational offerings. Whether you're look	{"loc":{"lines":{"to":9,"from":9}}}	[-0.014854158,0.0087731
44	We also specialize in wedding invites and event collateral. From save-the-dates	{"loc":{"lines":{"to":11,"from":11}}}	[-0.015381508,-0.013431
45	Pamphlets and brochures are another popular service. We offer tri-fold, bi-fold,	{"loc":{"lines":{"to":13,"from":13}}}	[-0.025102898,0.000677
46	Our services are not just about the print, but the experience. We offer design cc	{"loc":{"lines":{"to":15,"from":15}}}	[-0.011508759,0.0003671
47	We also take bulk and corporate orders. If you're a business looking to print m	{"loc":{"lines":{"to":17,"from":17}}}	[-0.006337547,-0.025800
48	Quality is at the heart of everything we do. All our products go through a quali	{"loc":{"lines":{"to":19,"from":19}}}	[0.009686787,0.0051575
49	If you're unsure where to begin, just ask! Our Sales Assistant chatbot is equipp	{"loc":{"lines":{"to":21,"from":21}}}	[-0.015844453,-0.012817
50	At TheWallCo, we don't just print. We help you express, celebrate, promote, anc	{"loc":{"lines":{"to":23,"from":23}}}	[-0.022105718,0.0090361

A.4: Supabase vector database showing vector embeddings and the chunks stored. Here, the knowledge base of the fictional store called "TheWallCo" is stored. (The knowledge base is created using ChatGPT which has all the services/products TheWallCo provides).