Motiv

Motiv Assist

An AI-Powered Personalized Work Assistant

Department of Electrical and Software Engineering, Schulich School of Engineering

SCHULICH School of Engineering
UNIVERSITY OF CALGARY

The Team

Henrique Andras	**David San Kim**	**Kevin Araujo**
Project Manager, UI/UX Design & AI	Infrastructure & Backend	Frontend & Backend
Tien Dat Johny Do	**Ana Perrone**	**Isaiah Lemieux**
Infrastructure, Deployment & AI	UI/UX Design & Frontend	UI/UX Design & Backend
Clark Lai / Motiv	**Behrouz Far**	**Mehdi Marzban**
Sponsor Representative	Academic Advisor	Teaching Assistant

Motivation and the Problem

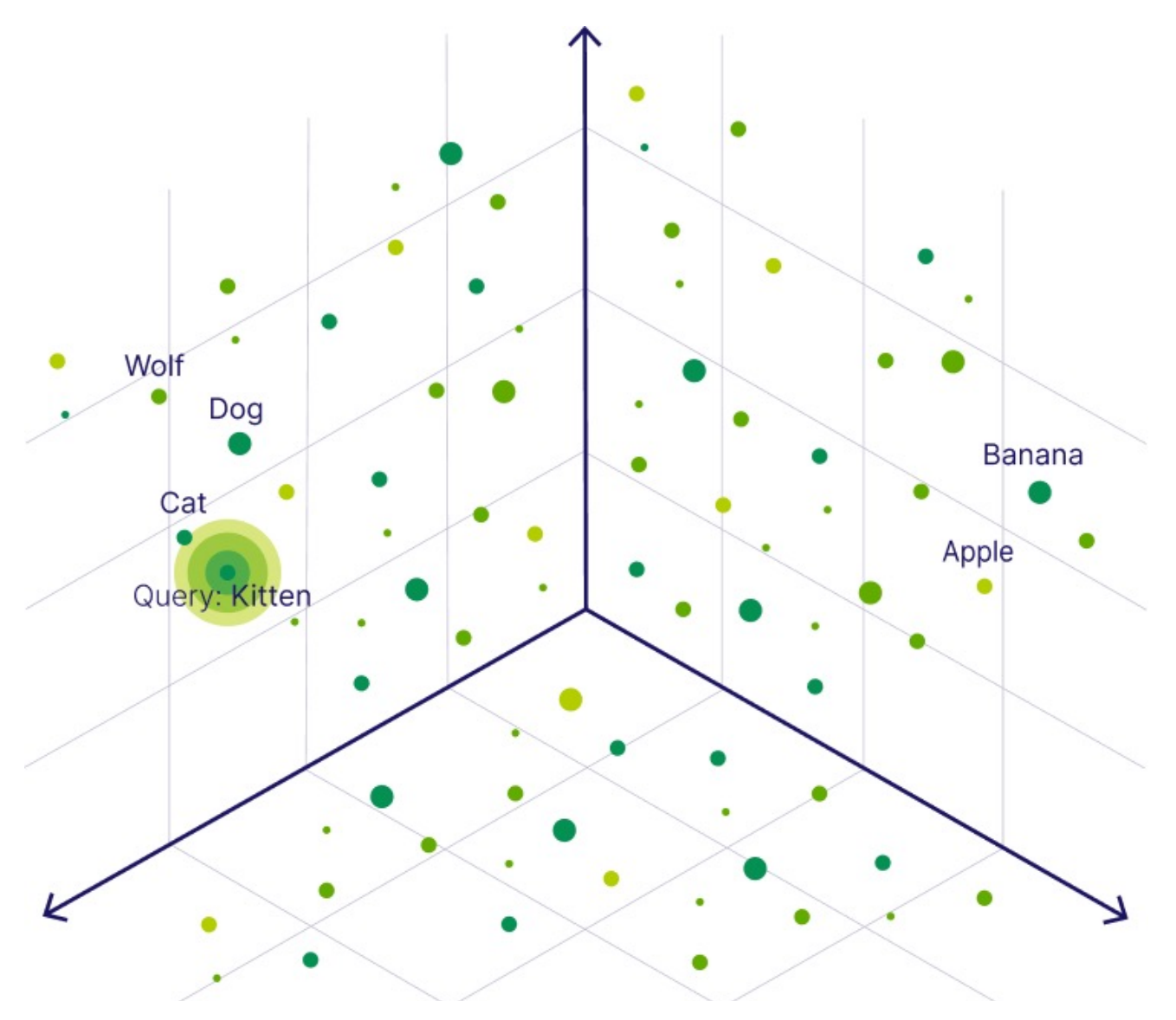
Businesses of all types store proprietary data on productivity platforms and web services that keep company information centralized and easily accessible to all members of the workforce. One such data source is Notion, a versatile organizational tool that offers use cases including documentation, task management, collaborative workspaces, client and sales management and more. However, while the collaborative abilities Notion and other productivity services provide make the business landscape more convenient than ever, these platforms still lack concrete personalization features that could enhance data interactions making them smarter and more efficient. This presents a significant problem in the dynamic business sector where every second counts. Employees must dig through hundreds of document pages to find small and large amounts of information necessary for the completion of trivial tasks, wasting valuable time in the process.

Objective and Project Scope

Our primary objective is to amplify the value derived from companies' proprietary data assets, while also enhancing the user experience within the context of conversational AI-powered data retrieval. Our solution entails constructing a personalized chatbot that seamlessly integrates with Notion, securely embedding records to be quickly and easily accessed by the automated assistant, enabling employees to engage in conversations with their own documents. This work assistance chatbot will tailor its responses, ensuring effortless access to companies' internal knowledge bases, all while prioritizing data security. An intuitive user interface will make interactions smoother, and the chatbot will incorporate memory and history-tracking features to foster context-aware and cohesive dialogues with users. Our goal is straightforward: Enabling employees to ask questions about their own documents and receive answers that truly suit their needs, making chatbots more helpful while keeping companies' data safe.

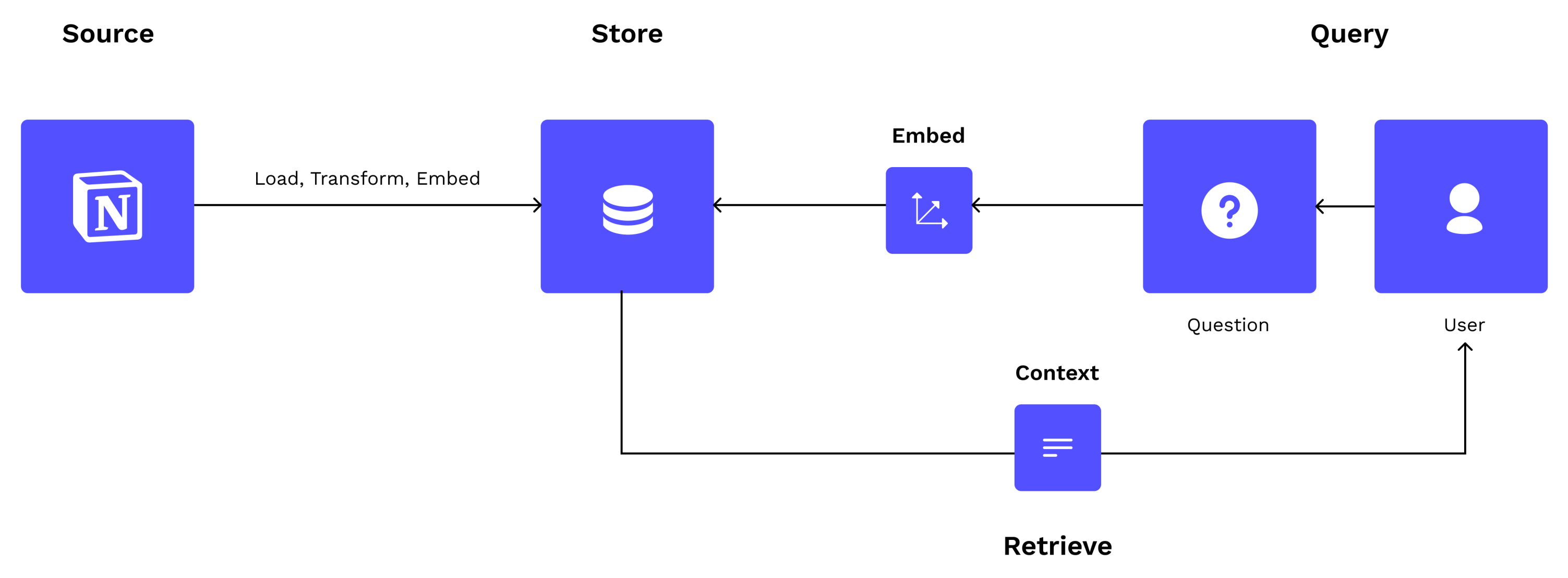
Embedding and Retrieval Process

Embedding and retrieval works based on a Retrieval-Augmented Generation (RAG) system which enriches a user's question with additional context from other sources to supply the retrieval algorithm creating highly relevant responses. Before embedding is commenced, documents from Notion must be extracted and converted into single page documents using LangChain. These documents are then split into small chunks of data using LangChain for the embedding. Embedding is completed by transforming text into a list of numbers allowing computers to understand, while preserving meaning and relationships between words in a text. This is accomplished using OpenAI's embedding model. These large documents, now split into small chunks of mathematical data, or vectors, are stored in a vector database, which plots each vector in close proximity to other semantically related vectors (simplified example below). For this project we used pgvector, a vector similarity search extension for PostgreSQL. The closer two vectors are in the multi-dimensional vector space, the stronger their semantic relationship is.



LangChain
OpenAI
PostgreSQL

Once the data embedding is completed, a user may ask a question. Like the data in the embedding step, the question itself must be embedded into the vector space to perform a similarity search. Then *k* vectors with the closest proximity to the user's question are collected, where *k* is defined by us, and is all sent to the Large Language Model (LLM) as "context". The LLM compiles the data and generates text providing an answer with relevant information back to the user.



Methodology

Throughout the duration of the project, our group employed the agile approach for managing project deliverables and tasks. This was completed using one-week scrums. During the initial design phase in the fall semester, tasks would be divided amongst group members to be completed or progressed through during the week. Depending on the complexity of a task at hand, sub-groups given a specific purpose could be formed. Weekly meetings held, were meant as a time to update the rest of the group on each task's progress, keeping all members up to date on specifications of the project. This ensured each group member possessed a maximum amount of understanding on project details that they had not worked on, which was crucial for the implementation stage during the winter semester.

For implementation, a list of features/requirements expected for the final product the group was visualizing was created, separated into unique levels of priority. Basic functionality was given the highest priority, while additional features fell to lower priorities depending on their importance and determined ease of implementation. This feature list was used to distinguish tasks during the implementation phase of the project.

For the winter semester, again using a one-week scrum cycle, two meetings were held per week. The first meant for minor updates to tasks or requesting support from other members, and the second meant to demo any new functionality completed during the week. This was repeated until the eventual completion of the project.

Conclusion

The group has been able to complete a final product which successfully integrates with Notion to process user's questions and return a context-aware response acting as a digital assistant with access to embedded files as selected by the user. This meets the initial goals of the project, allowing users to quickly acquire information, summaries, tasks, and more, saving hours searching through documents manually.

References

"Retrieval-augmented generation for knowledge-intensive NLP tasks." IBM Research Blog. Retrieved November 14, 2022, from https://research.ibm.com/blog/retrieval-augmented-generation-RAG

"Embeddings - OpenAI API." OpenAI. Retrieved November 14, 2022, from https://platform.openai.com/docs/guides/embeddings

"pgvector and Embedding Solutions with Postgres." Tembo. Retrieved November 14,
2022, from https://tembo.io/blog/pgvector-and-embedding-solutions-with-postgres/