RICOH GenAI Semantic Search

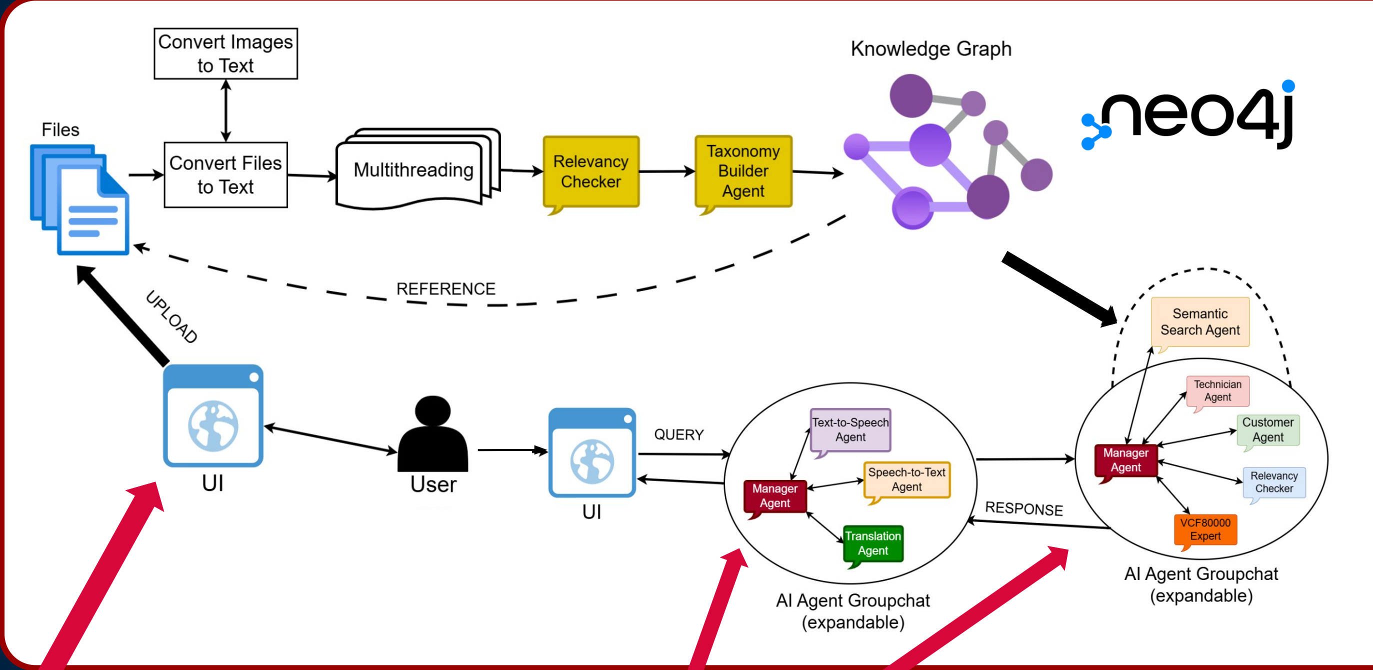
JJ Alles Cristian Madrazo Grace Luka Adam Krasovec

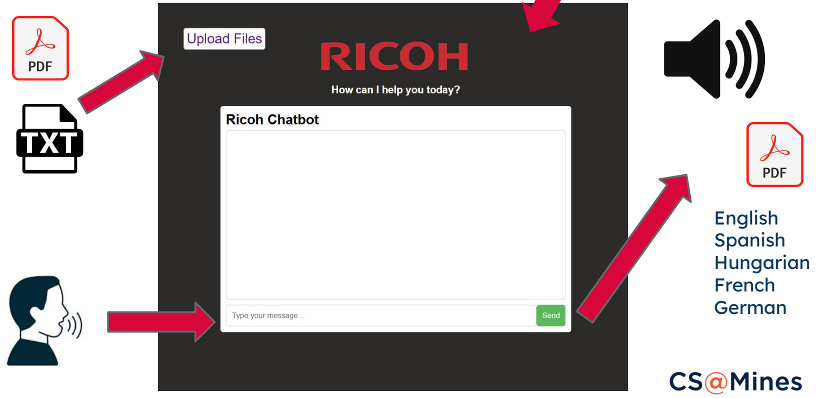
SCAN ME

COLORADO SCHOOL OF MINES

Introduction

The ultimate goal of this project was to provide Ricoh USA with a generative AI model that could ingest documents and provide contextual responses. Supporting this endeavor, our design needed to be fast, consistent, and easily understood.





English
Spanish
Hungarian
French
German

CS@Mines

Agentic Workflow + OpenAI Agents

An agentic workflow involves using AI agents to perform tasks autonomously by breaking them into smaller, manageable actions. These agents can collaborate, communicate, and execute specific functions to achieve a larger goal, working within predefined parameters or learning dynamically to adapt to challenges. In this workflow, each agent has a distinct role or expertise, and their interactions are orchestrated to ensure efficient task execution and coherent results.

Graph - Retrieval Augmented Generation

Graph Retrieval-Augmented Generation (Graph RAG) is a technique that combines knowledge graphs and generative AI to improve information retrieval and text generation tasks. It leverages the structured nature of knowledge graphs, where nodes represent entities and edges capture their relationships, to provide richer context and more precise retrieval compared to flat text-based systems. In this approach, a user's query is encoded and used to traverse the graph, retrieving relevant nodes and their relationships. The retrieved information is then fed into a generative model, such as GPT

The ultimate goal of this project was to provide Ricoh USA with a generative AI model that could ingest documents and provide contextual responses. Supporting this endeavor, our design needed to be fast, consistent, and easily understood.

Implementation of a chatbot GUI for users to interact with LLM

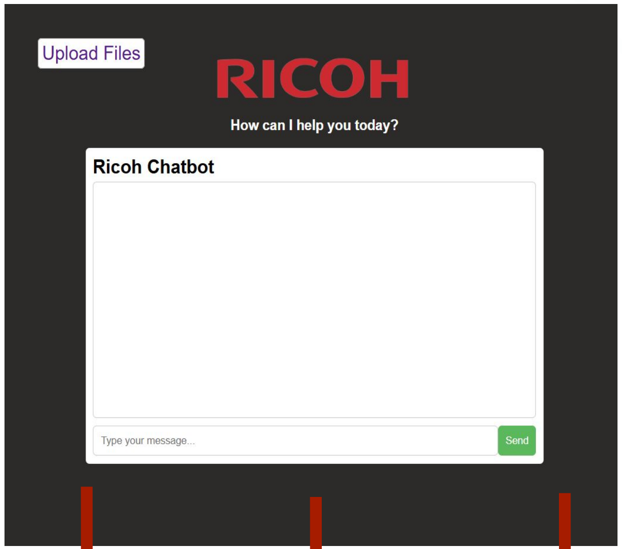
- GUI must be accessible via internal website
- Chatbot must be able to accept text input
- Chatbot must be able to accept speech input
- Chatbot must be able to accept input in multiple languages
- Chatbot must ask user for feedback on results
    - Chatbot feedback must be stored in a database

Chatbot backend must be based on open or closed-source pre-trained LLM

- LLM must be customized to understand semantics in input
- LLM must be able to retrieve and generate output from Ricoh's IP database
    - LLM's retrieval feature must be able to parse different data formats including html, pdf, MS Word, and various audio/video formats.
- There must be a user interface by which Ricoh's technicians can interact, submit inquiries, and get responses. A foundation for this will be supplied from previous semesters, but various changes may be required as the project evolves.

PDF and TXT files can be uploaded to add context

Speech to text available for inquiries



Multilingual application in the following languages:

English
French
Hungarian
Spanish
German

Chatbot returns references and resources in PDF

Text to speech available for responses

Cristian Madrazo
Denver, Colorado
cristian_madrazo@mines.edu
hecatommyriagon
cristian-madrazo

Grace Luka
Roxborough, Colorado
graceluka@mines.edu
grace-luka
https://grace-luka.github.io/#

JJ Alles
Happy Valley, Oregon
jacoballes@mines.edu
jacoballes
jacob-alles

Adam Krasovec
Budapest, Hungary
krasovecadam@mines.edu
krasoadam35
adam-krasovec

RICOH

GenAI Semantic Search

COLORADO SCHOOL OF MINES

CS@Mines

Docking

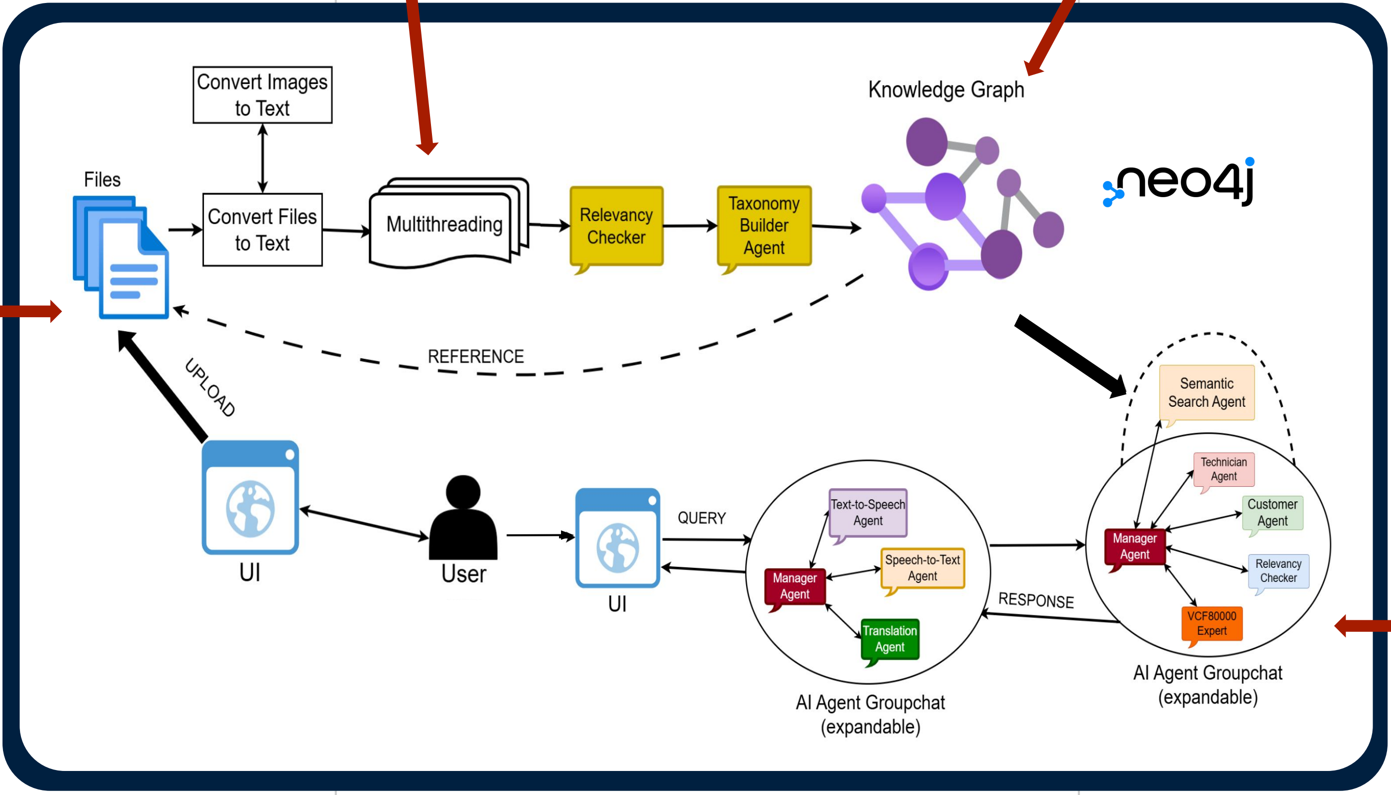
To ensure repeatability across devices, our team set up our code to run in Docker. This enables us to have a uniform development environment across different file systems. Additionally, we purposefully used widely supported Python libraries so as to ensure it's available to many different operating systems and browsers, with it currently opening the HTML in your default browser. Much of the heavy lifting is done by those libraries and APIs, meaning the code was consistent in our tests.

Uploading

When uploading a document serially, we determined that a certain document our client would provide would take upwards of twenty-five hours to fully process through OpenAI's tier 1 API. Instead, we determined that OpenAI's token limit could be utilized more fully, and with multithreading we were able to have fine control over the timing, limited only by OpenAI's TPM calls. The same document that would have taken more than a day could now be completed in less than three. This same multithreading approach works also for uploading directly to Neo4j. While Neo4j processes similarly to OpenAI, the document is immediately put into a usable knowledge graph and is immediately available for searches and queries. Additional processing is possible before feeding into the knowledge graph, as is discussed in future work. It was also calculated that OpenAI document processing could again be multiplied tenfold by spending more money on OpenAI's second tier. While OpenAI document processing wasn't strictly necessary it could greatly assist in categorizing, propositioning, and cleaning text before it goes into the database.

Multithreading

Performance was critical to our design, as we're dealing with parsing and uploading many chunks of text to APIs and databases. We iterated through a number of designs, and through many tests determined the best design was our multithreaded approach to API calls.



Graph - RAG

Graph Retrieval-Augmented Generation (Graph RAG) is a technique that combines knowledge graphs and generative AI to improve information retrieval and text generation tasks. It leverages the structured nature of knowledge graphs, where nodes represent entities and edges capture their relationships, to provide richer context and more precise retrieval compared to flat text-based systems. In this approach, a user's query is encoded and used to traverse the graph, retrieving relevant nodes and their relationships. The retrieved information is then fed into a generative model, such as. GPT, to produce a response that incorporates both the original query and the graph-derived context. The method's strengths lie in its ability to improve contextual understanding by utilizing entity relationships and ensuring retrieval precision by focusing on graph-relevant information. It is also dynamic and scalable, making it well-suited for large, evolving datasets such as organizational knowledge bases or scientific databases. For our project this semester, we used neo4j as our host for our knowledge graphs as per request from our client.

Agentic Workflow + Open AI Agents

An agentic workflow involves using AI agents to perform tasks autonomously by breaking them into smaller, manageable actions. These agents can collaborate, communicate, and execute specific functions to achieve a larger goal, working within predefined parameters or learning dynamically to adapt to challenges. In this workflow, each agent has a distinct role or expertise, and their interactions are orchestrated to ensure efficient task execution and coherent results. For our project, we chose the group chat pattern of agents because it mirrors how humans collaborate in team discussions, making the system intuitive and efficient. In this pattern, agents act as participants in a virtual group chat, exchanging information and reasoning collectively. This approach allows for flexible problem-solving as agents contribute their specialized knowledge, debate, and refine solutions in real-time. The group chat pattern also enables scalability, as new agents with additional expertise can be seamlessly added to the conversation.

RICOH

neo4j

OpenAI

python™

ChatGPT