Université Internationale de Rabat
L'UNIVERSITÉ INNOVANTE

Ecole Supérieure d'Informatique et du Numérique
COLLEGE OF ENGINEERING & ARCHITECTURE

Level : 4th year

Artificial Intelligence

CAREER ADVISOR CHATBOT

Encadré par :

Mr. GAMOUH Hamza & HAFIDI hakim

Réalisé par :

SEBAI Mohamed Zine El Abidine
KACHMAR Mohamed
EL OUAILI Hamza
AIT TAZALINE Aya

Academic year : 2024 - 2025

Table of contents

Abstract

The rapid expansion of data in diverse formats, including structured and unstructured sources, has posed significant challenges for effective information retrieval and decision-making. Traditional methods often fail to deliver relevant results when dealing with complex queries or multi-modal data. This project addresses these challenges by proposing a Hybrid Retrieval-Augmented Generation (RAG) system that integrates Neo4j for graph-based relational data queries and ChromaDB for semantic vector-based searches. By leveraging Large Language Models (LLMs) such as Gemini Pro, the system enhances retrieval accuracy and generates context-aware responses, bridging the gap between structured and unstructured data processing.

The implemented solution includes a Gradio-powered user interface, providing an intuitive platform for users to interact with the system. This hybrid architecture demonstrates the capability to answer complex queries by combining graph-based relationships with semantic similarity search, making it a powerful tool for diverse applications. Key contributions include the development of a unified pipeline, exploration of hybrid RAG architectures, and insights into optimizing user experience for real-world scenarios. The findings underscore the potential of hybrid systems in transforming information retrieval processes and set the stage for further innovations in AI-driven data analysis.

Keywords

- **Retrieval-Augmented Generation (RAG):** A hybrid approach combining traditional and vector-based retrieval methods with generative capabilities to enhance information retrieval.
- **Semantic Search:** A technique that uses embeddings to retrieve contextually relevant data based on meaning rather than keywords.
- **Hybrid Architecture:** An integrated system combining multiple retrieval techniques for enhanced query handling and response accuracy.
- **Vector Embeddings:** Numeric representations of textual data enabling semantic similarity comparisons.

Abbreviations and Definitions

Term	Definition
RAG	Retrieval Augmented Generation, an architecture combining information retrieval and text generation to provide enriched responses based on documents (Lewis et al., 2020).
Neo4j	A graph database that stores and queries complex relationships between entities in graph format (Neo4j Documentation, 2024).
ChromaDB	An in-memory vector database designed for semantic similarity searches in embeddings, used for semantic search tasks (ChromaDB Team, 2024).
LangChain	A framework for building LLM pipelines, integrating retrieval, analysis, and generation steps (LangChain, 2024).
LLM	Large Language Model, an AI system that generates text based on instructions or prompts with near-human performance (Naveed et al., 2024).
Embedding	A numeric representation of a document or phrase in a vector space, enabling semantic similarity computations (Mikolov et al., 2013).
Gradio	An open-source framework for designing interactive user interfaces for machine learning applications.

Prompt	An instruction or question provided to an LLM to generate a response.
Gemini Pro	An advanced language model API used to provide responses based on extracted contexts (Gemini, 2024).

1.Introduction

1.1 Problem Description

In the era of rapid digitalization, the volume of data generated daily is growing exponentially, presenting significant opportunities and challenges in information retrieval and decision-making. This project aims to address the limitations of conventional retrieval systems, which often struggle with the seamless integration of structured and unstructured data. While structured data—such as relational databases—provides clear, interconnected relationships, unstructured data (e.g., text, images) offers immense potential for deeper insights when processed effectively. However, retrieving and synthesizing both types of data within a single query remains a persistent challenge.

To bridge this gap, this project introduces a **Hybrid Retrieval-Augmented Generation (RAG) system**, combining the capabilities of graph-based relational databases and vector-based semantic search systems. Specifically, the system leverages **Neo4j**, a graph database known for its ability to model and query complex relationships, and **ChromaDB**, a vector storage solution designed for semantic similarity searches. Together, these components are orchestrated to retrieve, integrate, and contextualize data, delivering highly relevant results.

At the core of this hybrid system is the **Gemini Pro**, a state-of-the-art Large Language Model (LLM) that synthesizes retrieved data into coherent and contextually accurate responses. Users interact with the system through an intuitive **Gradio-powered user interface**, enabling them to query both structured and unstructured data seamlessly. This approach provides a practical solution for addressing complex, multi-modal queries while offering a user-friendly interface accessible to technical and non-technical users alike.

1.2 Objective

The objective of this project is to design and implement a Hybrid Retrieval-Augmented Generation (RAG) system that bridges the gap between structured and unstructured data retrieval. This system integrates Neo4j for structured graph-based queries and ChromaDB for embedding-based similarity searches, creating a unified pipeline for robust information retrieval. By incorporating advanced Large Language Models (LLMs) like Gemini Pro, the system enhances the ability to interpret queries, retrieve relevant data, and generate comprehensive, context-aware responses.

The project also aims to provide a user-friendly interface powered by Gradio, enabling seamless interaction between users and the system. This intuitive interface will allow users to input complex queries and receive meaningful results, making the system accessible to non-technical users while maintaining the flexibility required for developers to customize and extend its functionalities.

1.3 Research Questions

This study aims to address the following research questions:

1. How can a hybrid RAG system improve the retrieval accuracy and contextual relevance of responses from diverse data sources?
2. What are the technical and architectural benefits of combining Neo4j for structured data retrieval with ChromaDB for vector-based search?
3. How can advanced LLMs, such as Gemini Pro, be leveraged to enhance the generation of coherent and contextually appropriate responses?
4. What design principles can maximize usability and accessibility in the Gradio-powered user interface for hybrid RAG systems?

1.4 Key Contributions

1. **System Design:** Implementation of a hybrid pipeline integrating Neo4j, ChromaDB, and Gemini Pro to unify structured and unstructured data retrieval.
2. **Workflow Automation:** Automating query generation and data fusion through prompt engineering and vector embeddings.
3. **Practical Demonstration:** Real-world use cases, such as CV analysis and career recommendations, showcasing the system's effectiveness in addressing user-specific queries.
4. **Scalability and Usability:** Ensuring that the system remains robust under varying data sizes while maintaining a user-friendly experience.

This introduction sets the foundation for understanding the project's scope, methodology, and potential impact, guiding the reader through the subsequent sections detailing its design, implementation, and results.

1.5 Scope and Limitations

Scope

The scope of this project includes:

- Implementing a hybrid RAG system capable of retrieving and synthesizing information from structured and unstructured data sources.
- Integrating Neo4j for querying graph-based structured data and ChromaDB for vector similarity search.
- Utilizing Gemini Pro, a cutting-edge LLM, to interpret user queries and generate human-like responses.
- Developing a Gradio-based user interface to facilitate real-time interaction and visualization of results.
- Demonstrating the system's capability through case studies or practical scenarios involving complex, multi-modal queries.

Limitations

While the project presents an innovative approach, several limitations must be acknowledged:

- The system's performance depends on the quality and comprehensiveness of the underlying data sources and embeddings.
- High computational costs associated with LLMs like Gemini Pro may limit scalability for large datasets or real-time processing.
- The retrieval accuracy is constrained by the limitations of the Neo4j schema and the embedding models used in ChromaDB.
- The system currently does not support real-time updates in data sources, requiring manual re-indexing to ensure up-to-date retrieval.

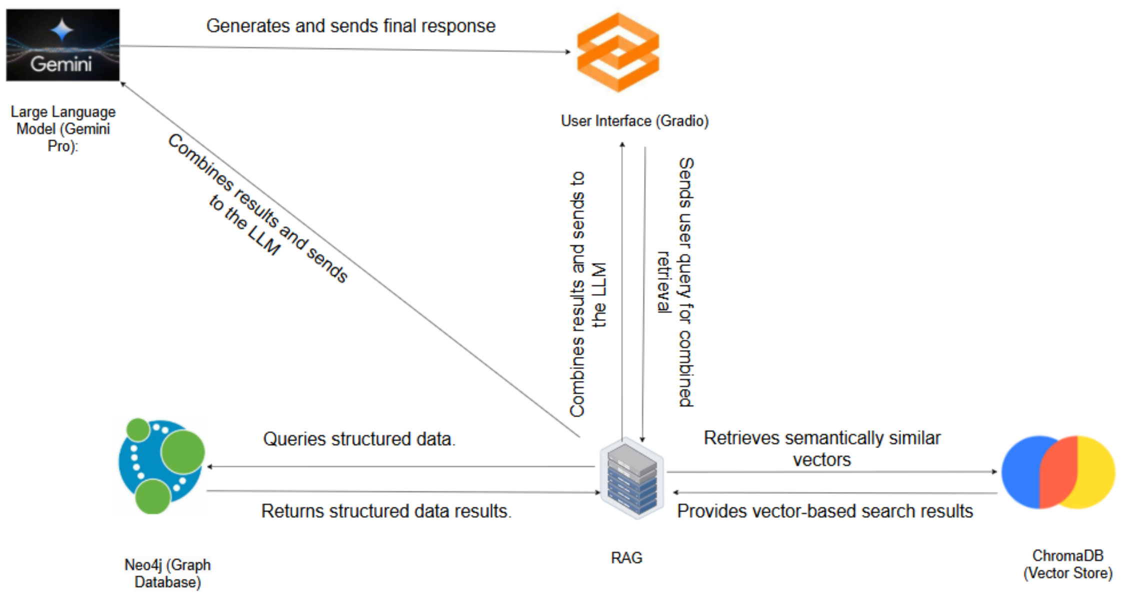
2. System Design and Methodology

2.1 Overview of the System Architecture

This section provides a comprehensive overview of the system architecture designed for this project. The architecture integrates the following components:

- **Neo4j (Graph Database):** Used to store and query structured, relational data, providing contextual information based on graph relationships.
- **ChromaDB (Vector Store):** Serves as the vector-based storage, enabling semantic similarity search across unstructured data.
- **Large Language Model (Gemini Pro):** Responsible for synthesizing information retrieved from both Neo4j and ChromaDB to generate coherent and contextually accurate responses.
- **User Interface (Gradio):** Acts as the entry point for users to interact with the system, providing an intuitive platform for submitting queries and receiving results.

A detailed architecture diagram is provided to illustrate the relationships and data flow between these components:


Gemini
Large Language Model (Gemini Pro):
Generates and sends final response
User Interface (Gradio)
Combines results and sends to the LLM
Combines results and sends to the LLM
Sends user query for combined retrieval
Queries structured data.
Returns structured data results.
Retrieves semantically similar vectors
Provides vector-based search results
Neo4j (Graph Database)
RAG
ChromaDB (Vector Store)

Global Architecture

User Interface (Gradio) → RAG: Sends user query for combined retrieval.

RAG → Neo4j (Graph Database): Queries structured data.

RAG → ChromaDB (Vector Store): Retrieves semantically similar vectors.

Neo4j (Graph Database) → RAG: Returns structured data results.

ChromaDB (Vector Store) → RAG: Provides vector-based search results.

RAG → Large Language Model (Gemini Pro): Combines results and sends to the LLM.

Large Language Model (Gemini Pro) → User Interface (Gradio): Sends the final response to the user.

2.2 Technologies, Tools, and Implementation

This section details the technologies, tools, and implementation processes used in the project.

Neo4j

- **Role in the System:** Used for structured data storage and retrieval, enabling graph-based relational queries.
- **Schema Design:** Nodes represent entities, and edges represent relationships, providing a flexible and context-rich data model.
- **Query Optimization:** Cypher queries are optimized to ensure fast and relevant responses based on user requirements.

ChromaDB

- **Role in the System:** Serves as the vector store for unstructured data, enabling semantic similarity searches.
- **Embedding Generation:** Textual data is converted into vector embeddings using pre-trained models.

- **Storage and Retrieval:** Indexed embeddings allow for efficient and accurate retrieval of relevant unstructured data.

LangChain

- **Integration:** Acts as the orchestrator of the RAG pipeline, managing interactions between Neo4j, ChromaDB, and Gemini Pro.
- **Customization:** Facilitates the design of modular workflows for complex query handling.

Gemini Pro

- **Role in the System:** Processes and synthesizes the data retrieved from Neo4j and ChromaDB to generate context-aware responses.
- **Prompt Design:** Custom prompts ensure the generation of relevant, coherent answers tailored to user queries.
- **API Integration:** Seamless interaction with Neo4j and ChromaDB data for final response construction.

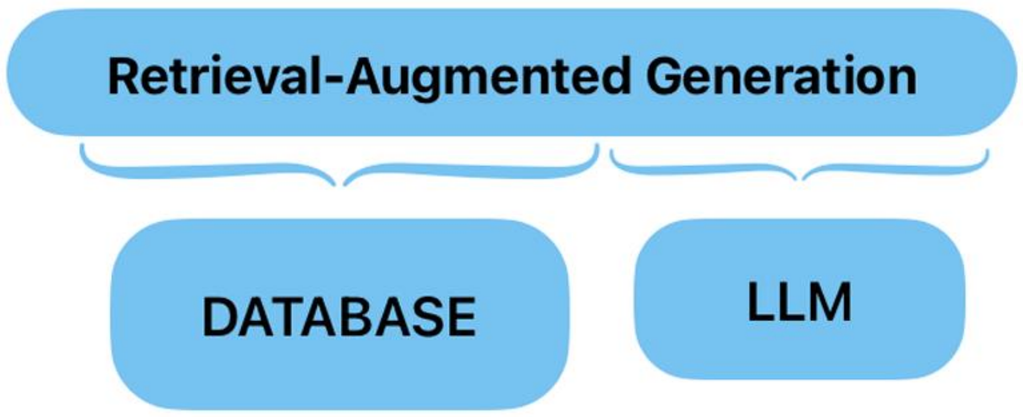
Gradio

- **Role in the System:** Provides the user interface for query submission and result display.
- **Interface Design:** A clean, intuitive web-based platform for end-user interactions.
- **Backend Integration:** Manages query submission and integrates with the RAG pipeline to display responses.

gradio

2.3 Hybrid RAG Workflow

The RAG approach combines retrieval-based and generative models to enhance the quality of generated text in natural language processing and machine learning tasks. By retrieving relevant information from a database and using it to inform text generation – RAG produces more accurate, informative, and contextually relevant outputs and reduces hallucinations. RAG is especially useful if a model needs to give up-to-date or domain-specific information (Miao et al., 2024). Figure 1 below depicts which part of the RAG architecture is retrieving, and which part is generating. 10 RAG offers an alternative to fine-tuning for adapting LLMs to domain-specific tasks. Fine-tuning updates the model's weights using a predefined training set, like questions and answers, to improve accuracy (Miao et al., 2024). In contrast, RAG uses a retriever to fetch data and provide it as context to the LLMs prompt. This approach is less resource-intensive and allows easier updates. However, RAG's dependency on context limits the amount of information provided per prompt, based on the LLM context window.



"Simplified visual of the RAG architecture. Showcasing the idea of RAG retrieving relevant information from a database and generating responses via text generating models i.e. an LLM."

The Hybrid Retrieval-Augmented Generation (RAG) pipeline functions as follows:

1. **User Query Submission:** The user inputs a query through the Gradio interface.
2. **Neo4j Query:** The RAG module queries Neo4j to retrieve structured, relational data relevant to the query.
3. **ChromaDB Search:** Simultaneously, the RAG module searches ChromaDB for semantically similar embeddings, retrieving relevant unstructured data.
4. **Data Fusion:** Results from Neo4j and ChromaDB are combined and sent to Gemini Pro for further processing.
5. **Response Generation:** Gemini Pro synthesizes the retrieved data and generates a comprehensive response.
6. **Result Display:** The final response is displayed to the user through the Gradio interface.

2.4 Large Language Model

LLMs are advanced AI systems specialized in text processing and generation. LLMs include text generation, image generation and embeddings. They perform exceptionally well in natural language tasks, intriguing significant research interest. While LLMs generate coherent responses to task queries based on descriptions and examples, Naveed et al. (2024) acknowledge the challenges in understanding user intent – together with noticing worse performance in zero-shot learning and better performance in few-shot learning. Few-shot learning is a prompt engineering technique where input-output demonstrations are provided to the model.

LLMs may struggle with limited memory and outdated information resulting in inaccurate responses. With information retrieval techniques LLMs performs with more accurate answers, references and can access more information. Models with implemented retrieval augmentation have been shown to perform better (Naveed, et al., 2024).

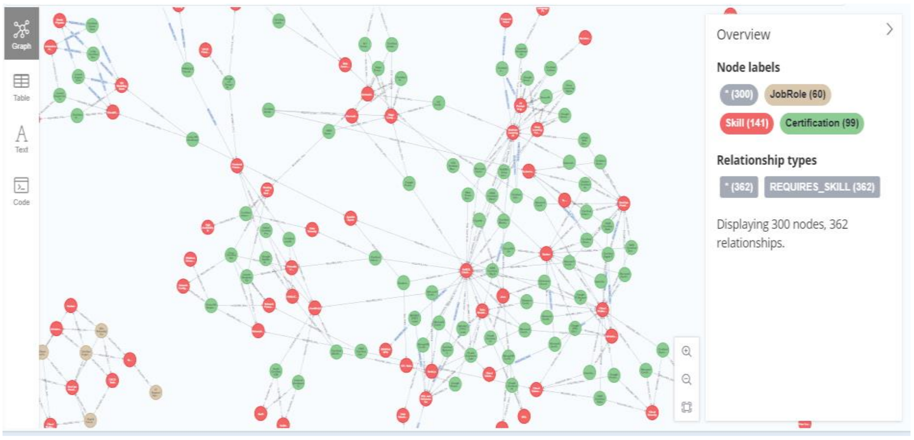
Generally, interacting with an LLM involves feeding the model with words, and the model returns words as output either be trying to complete the sentence or trying to answer a provided question. An LLM processes these words by tokenizing them, which involves decomposing them into smaller components

(Jeong & Kyoung Jun Lee, 2024). For example, the sentence: “I want to sharpen my competencies”, containing 5 words gets decomposed into 7 smaller tokens where competencies are divided into “compet” and “encies”. An important part to consider when using an LLM, is the maximum number of tokens, also known as context window. This is the maximum number of tokens the model can take as input together with the number of tokens that can be provided as output.

3. Results and Analysis

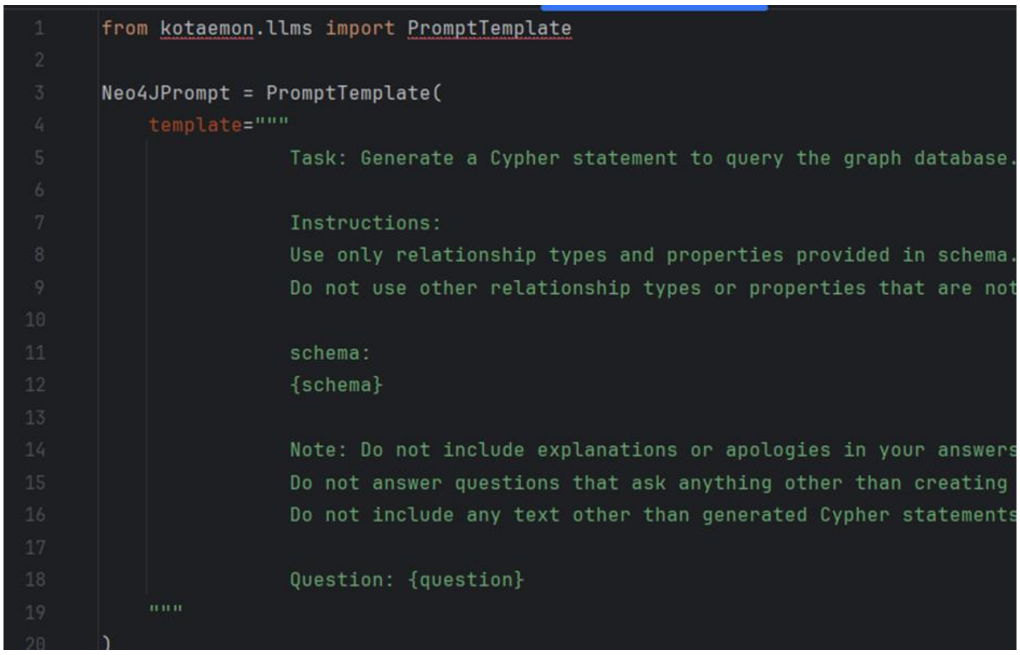
3.1 Neo4j Query Construction

- **Description:** The Neo4j schema is utilized to craft precise Cypher queries for structured data retrieval. This process ensures the extraction of contextually relevant data.



"Neo4j schema used to generate Cypher queries for accurate data retrieval."

- **Description:** This code snippet demonstrates how prompts are crafted using Neo4j's schema to generate Cypher statements. The template strictly adheres to the provided schema and generates accurate queries for retrieving structured data.

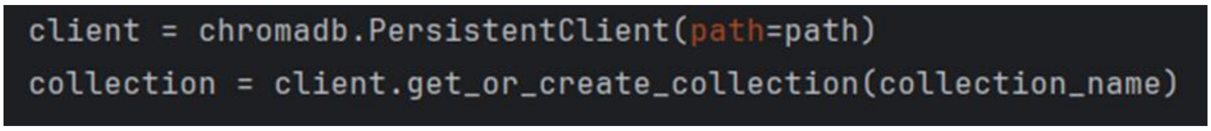


```
from kotaemon.llms import PromptTemplate

Neo4JPrompt = PromptTemplate(
    template="""
                Task: Generate a Cypher statement to query the graph database.

                Instructions:
                Use only relationship types and properties provided in schema.
                Do not use other relationship types or properties that are not

                schema:
                {schema}

                Note: Do not include explanations or apologies in your answers
                Do not answer questions that ask anything other than creating
                Do not include any text other than generated Cypher statements

                Question: {question}
    """
)
```

"Neo4j prompt template for generating Cypher statements to query the graph database."

3.2 ChromaDB Vector Store

- **Description:** ChromaDB is employed to create and manage vector stores, enabling efficient semantic similarity searches.

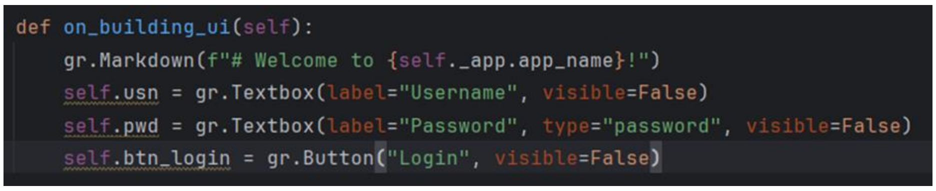


```
client = chromadb.PersistentClient(path=path)
collection = client.get_or_create_collection(collection_name)
```

"ChromaDB generating vector embeddings from documents for semantic search."

3.3 User Interface Implementation

- **Description:** The Gradio interface facilitates user interaction by providing input fields and buttons for seamless query submission and response retrieval. The example code demonstrates how the UI components, such as text boxes for username and password and a login button, are built using Gradio.

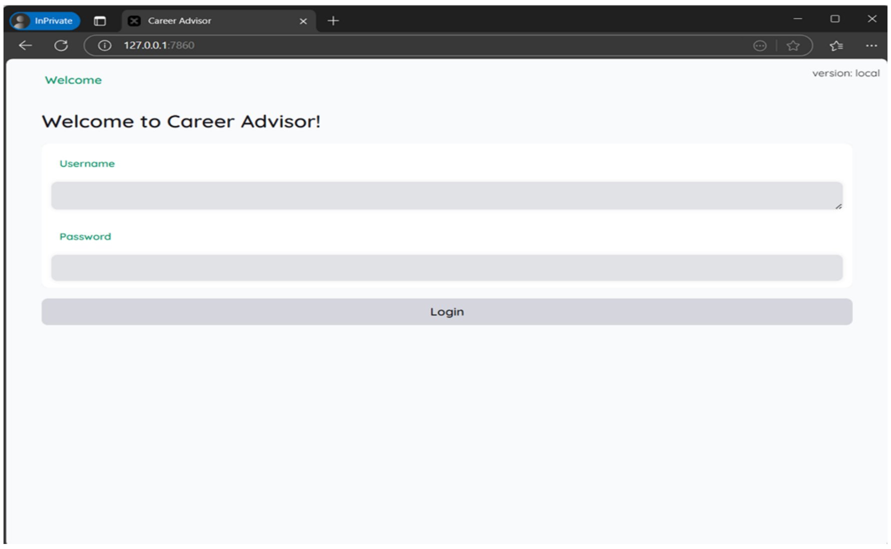


```
def on_building_ui(self):
    gr.Markdown(f"# Welcome to {self._app.app_name}!")
    self.usn = gr.Textbox(label="Username", visible=False)
    self.pwd = gr.Textbox(label="Password", type="password", visible=False)
    self.btn_login = gr.Button("Login", visible=False)
```

"Code snippet demonstrating the implementation of the Gradio-based user interface components."

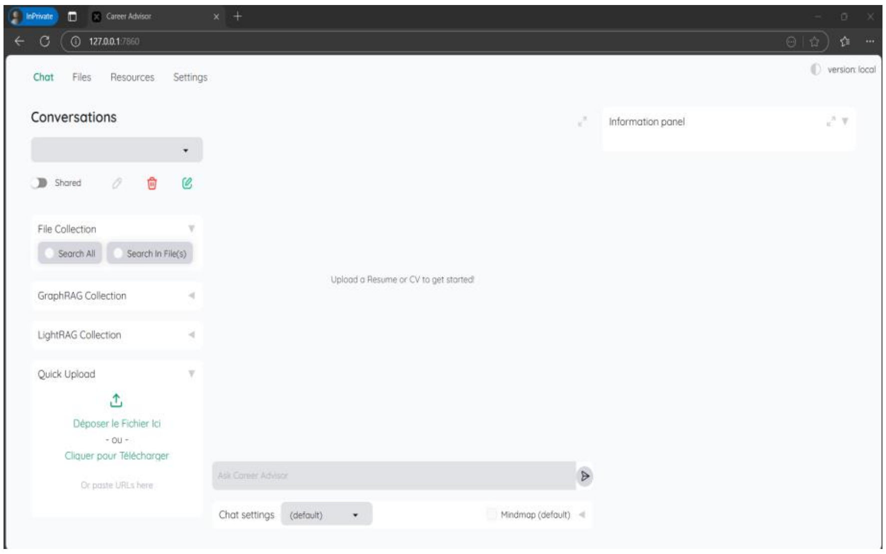
3.4 User Interface Implementation

- **Description:** The Gradio interface provides an intuitive layout for user interactions. This screenshot showcases the login screen where users can input their credentials. The interface emphasizes simplicity and ease of use for end-users.



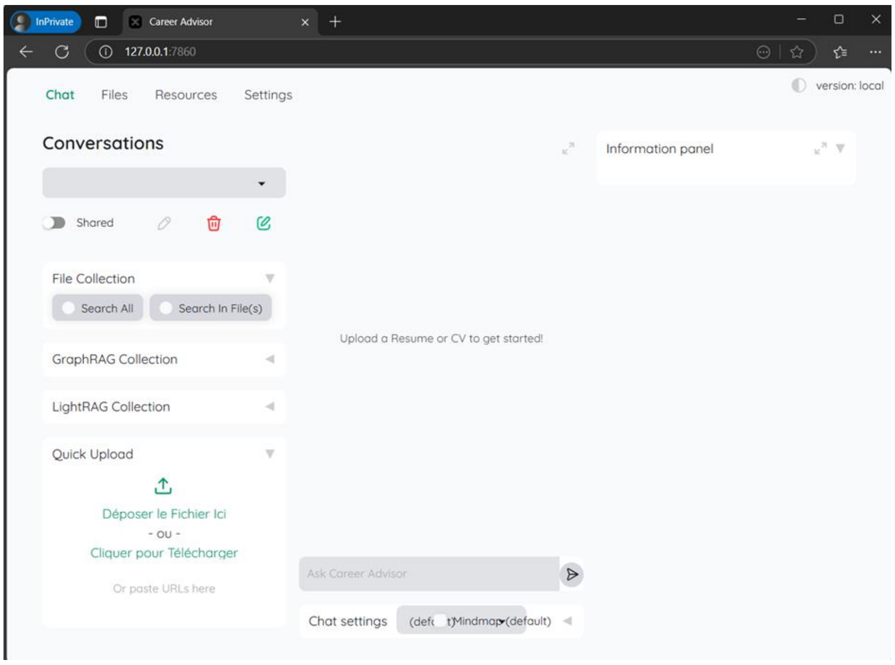
"Gradio-based user interface for query submission and login."

Description: This screenshot showcases the main screen where users can upload resumes or files, access file collections, and utilize the search functionality. The information panel dynamically updates based on the user's actions.



"Main interaction screen of the Gradio interface, showcasing file upload and search functionalities."

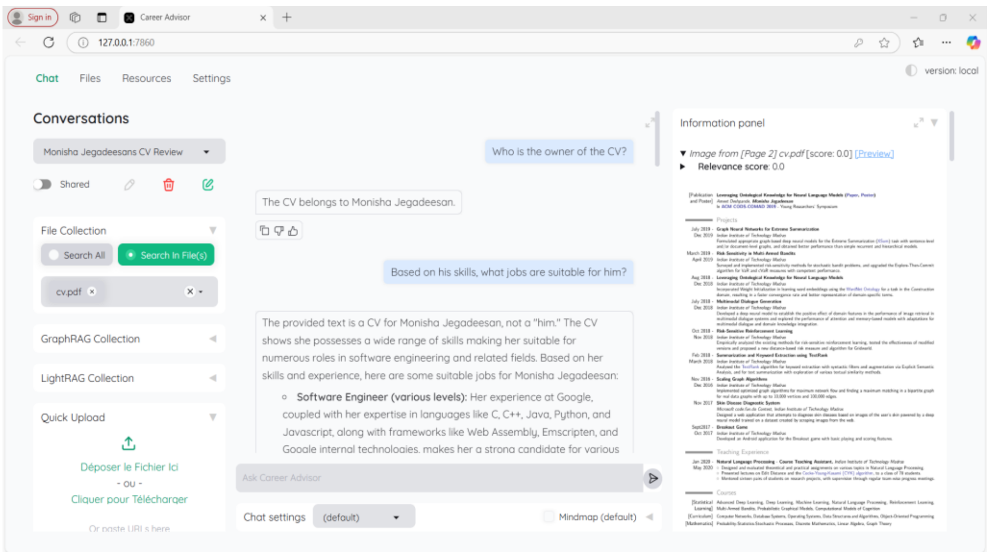
- **Description:** This interface screen provides users with options for uploading files, accessing different data collections, and performing searches. It serves as the central hub for interactions with the Career Advisor system.



"Main workspace of the Career Advisor Gradio interface, highlighting file upload and collection management features."

3.5 System Functionality Demonstration

- **Description:** The screenshot showcases the user querying the system and receiving responses based on the provided CV. The system retrieves and processes data using the Hybrid RAG approach, displaying relevant results in the information panel alongside the conversational interface.



Who is the owner of the CV?

The CV belongs to Monisha Jegadeesan.

Based on his skills, what jobs are suitable for him?

The provided text is a CV for Monisha Jegadeesan, not a "him." The CV shows she possesses a wide range of skills making her suitable for numerous roles in software engineering and related fields. Based on her skills and experience, here are some suitable jobs for Monisha Jegadeesan:

- **Software Engineer (various levels):** Her experience at Google, coupled with her expertise in languages like C, C++, Java, Python, and Javascript, along with frameworks like Web Assembly, Emscripten, and Google internal technologies, makes her a strong candidate for various

▼ *Image from [Page 2] cv.pdf* [score: 0.0] [Preview]
▶ Relevance score: 0.0

[Publication and Poster] **Leveraging Ontological Knowledge for Neural Language Models** (Paper, Poster)
Ameet Deshpande, **Monisha Jegadeesan**
In ACM CODS-COMAD 2019 - Young Researchers' Symposium

Projects

July 2019 - Dec 2019 **Graph Neural Networks for Extreme Summarization**
March 2019 - April 2019 **Risk Sensitivity in Multi Armed Bandits**
Aug 2018 - Dec 2018 **Leveraging Ontological Knowledge for Neural Language Models**
July 2018 - Dec 2018 **Multimodal Dialogue Generation**
Oct 2018 - Nov 2018 **Risk-Sensitive Reinforcement Learning**
Feb 2018 - March 2018 **Summarization and Keyword Extraction using TextRank**
Nov 2016 - Dec 2016 **Scaling Graph Algorithms**
Nov 2017 **Skin Disease Diagnostic System**
Sept2017 - Oct 2017 **Breakout Game**

Teaching Experience

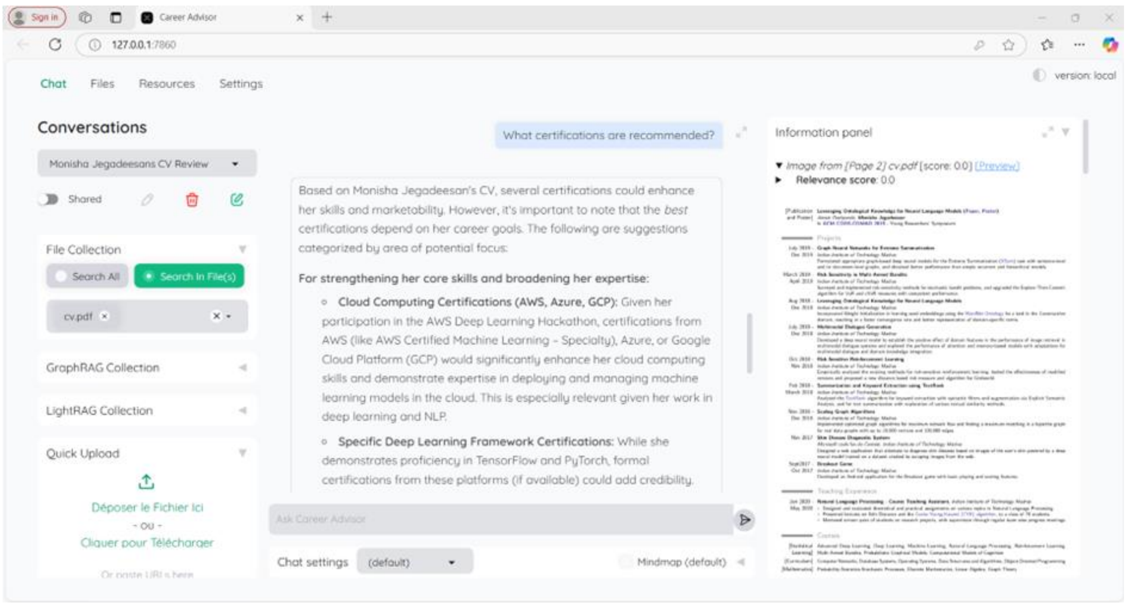
Jan 2020 - May 2020 **Natural Language Processing - Course Teaching Assistant**, Indian Institute of Technology Madras

Courses

[Statistical Learning] Advanced Deep Learning, Deep Learning, Machine Learning, Natural Language Processing, Reinforcement Learning, Multi-Armed Bandits, Probabilistic Graphical Models, Computational Models of Cognition
[Curriculum] Computer Networks, Database Systems, Operating Systems, Data Structures and Algorithms, Object-Oriented Programming
[Mathematics] Probability Statistics Stochastic Processes, Discrete Mathematics, Linear Algebra, Graph Theory

"Query and response interaction in the Career Advisor system, showcasing contextual answers and relevant document highlights."

- **Description:** This screenshot highlights the Career Advisor system's capability to recommend certifications based on a CV. The system uses the Hybrid RAG framework to analyze the CV and provide relevant recommendations for skill enhancement. The information panel displays additional context, ensuring transparency in the results.



What certifications are recommended?

Based on Monisha Jegadeesan's CV, several certifications could enhance her skills and marketability. However, it's important to note that the *best* certifications depend on her career goals. The following are suggestions categorized by area of potential focus:

For strengthening her core skills and broadening her expertise:

- **Cloud Computing Certifications (AWS, Azure, GCP):** Given her participation in the AWS Deep Learning Hackathon, certifications from AWS (like AWS Certified Machine Learning - Specialty), Azure, or Google Cloud Platform (GCP) would significantly enhance her cloud computing skills and demonstrate expertise in deploying and managing machine learning models in the cloud. This is especially relevant given her work in deep learning and NLP.
- **Specific Deep Learning Framework Certifications:** While she demonstrates proficiency in TensorFlow and PyTorch, formal certifications from these platforms (if available) could add credibility.

▼ *Image from [Page 2] cv.pdf* [score: 0.0] [Preview]
▶ Relevance score: 0.0

"System-generated certification recommendations based on CV analysis, highlighting relevant details in the information panel."

3.6 Summary

This chapter presented the results and analysis of the Hybrid RAG system. The integration of Neo4j and ChromaDB showcased the system's capability to retrieve and synthesize structured and unstructured data seamlessly. Through the Gradio-powered interface, the system provided an intuitive user experience while delivering contextually accurate responses. The real-world demonstrations highlighted the practicality and effectiveness of the system in handling complex queries. Challenges faced during implementation were also addressed, paving the way for future enhancements.

Conclusion

The development of the Hybrid Retrieval-Augmented Generation (RAG) system represents a significant milestone in addressing the challenges posed by integrating structured and unstructured data sources for advanced information retrieval. By leveraging the unique strengths of Neo4j for graph-based queries and ChromaDB for semantic vector searches, the system demonstrates the ability to provide highly relevant and context-aware responses to complex user queries.

A key innovation of the project lies in its integration with the state-of-the-art Large Language Model (Gemini Pro), which synthesizes data from diverse sources into coherent, user-friendly responses. The Gradio-powered interface further enhances the system's accessibility, enabling seamless interaction for a broad spectrum of users, from technical professionals to non-experts.

The project successfully demonstrates real-world applications such as CV analysis and career recommendations, showcasing the potential of hybrid AI systems in solving domain-specific challenges. Moreover, the implementation highlights how hybrid architectures can combine relational and semantic searches to improve both the accuracy and relevance of retrieved information.

Despite these accomplishments, the system is not without limitations. Challenges such as computational overhead, dependency on data quality, and the lack of real-time updates underscore areas for further research and development. Addressing these limitations could involve optimizing the integration pipeline, enhancing scalability, and extending support for multi-modal data types, including images and audio.

In conclusion, this project illustrates the transformative potential of hybrid AI architectures in redefining information retrieval processes. By bridging the gap between structured and unstructured data, the Hybrid RAG system sets a new benchmark for innovation in artificial intelligence and provides a strong foundation for future advancements in the field.