Proposal of ChatGIS: GIS-Powered Natural Language Query System

Group Name: ChatDB 70
Course: DSCI 551 - Foundations of Data Science
Instructor: Dr. Wensheng Wu
Institution: Viterbi School of Engineering, University of Southern California
Date: March 3, 2025

Team Members:
- [Yucheng Liu] - [Project Designing, Data Collection, Document Writing]

Team Members Background:

Yucheng Liu is a Master's student in Spatial Data Science at USC with a strong background in computer science, geospatial data science, and deep learning. He specializes in knowledge distillation, cloud-based application development, and geospatial analytics. Proficient in Python, SQL, NoSQL, and GIS, he has experience with AWS, TensorFlow, and ArcGIS Pro. His research focuses on knowledge distillation and multimodal learning in the physiological signals, with publications in ACM MM and IJCAI.

Project Abstract - This proposal presents a **GIS-powered natural language query system** called ChatGIS that integrates **PostGIS** and **LLMs** to allow users to retrieve geospatial data using natural language queries. The system translates user queries into **optimized SQL statements** that efficiently interact with a GIS database, enabling users to ask spatial questions such as *"Where are the nearest electric vehicle charging stations?"* or *"Find all Chinese restaurants within 5 km of my location."*. ChatGIS levels up the accessibility and usability of geospatial data by leveraging **LLM-based query processing, spatial indexing, and GIS visualization tools**.

Keywords: GIS, LLM, PostGIS, Natural Language Queries, Spatial Database, Geospatial Search, Route Optimization, OpenStreetMap, API Development, Spatial Indexing, Machine Learning, Location-Based Services

1 Introduction

Geospatial databases provide powerful spatial analysis capabilities but often require SQL expertise to query effectively. This make the GIS system hard to use for the non-technical user. To address this issue, we propose ChatGIS, a **GIS-powered natural language query system** that integrates **PostGIS** and **LLMs** to allow users to retrieve geospatial data using natural language queries. The system translates user queries into **optimized SQL statements** that efficiently interact with a GIS database, enabling users to ask spatial questions such as *"Where are the nearest electric vehicle charging stations?"* or *"Find all Chinese restaurants within 5 km of my location."*. ChatGIS levels up the accessibility and usability of geospatial data by leveraging **LLM-based query processing, spatial indexing, and GIS visualization tools**.

This project aims to integrate **Large Language Models (LLMs)** with **PostGIS**, enabling users to perform spatial queries in California using natural language. By translating user queries into SQL, the system facilitates intuitive geospatial searches and, if time permits, visualizes the results on the web. Additionally, we provide a **Flask API** to support seamless integration into third-party applications. Furthermore, we plan to explore cloud deployment to offer **Location-Based Services** to users, contingent on available time.

2 Data Source

In this project, we will use the **OpenStreetMap (OSM)** dataset as the primary data source. OSM is a collaborative project that collects and distributes geospatial data, and it is a popular choice for geospatial research and applications. OSM provides a rich set of features, including points of interest, roads, buildings, and natural features. OSM data is available in a variety of formats, including XML, PBF, and SHP. In this project, we will download the dataset in the California to build the GIS database.

3 Implementation

3.1 System Overview

The overview of the system is shown in Figure 1. The system consists of three main components: **Front-end** and **Back-end (including GIS database, LLM model, and server API)**. The front-end is responsible for user input, query processing, and visualization. The LLM model is responsible for translating the input into SQL statements, optimizing the queries, and feedbacking the results. The GIS database is responsible for storing and managing the geospatial data as well as the spatial analysis. The server API is responsible for providing a RESTful interface for the front-end and third-party applications to interact with the system.

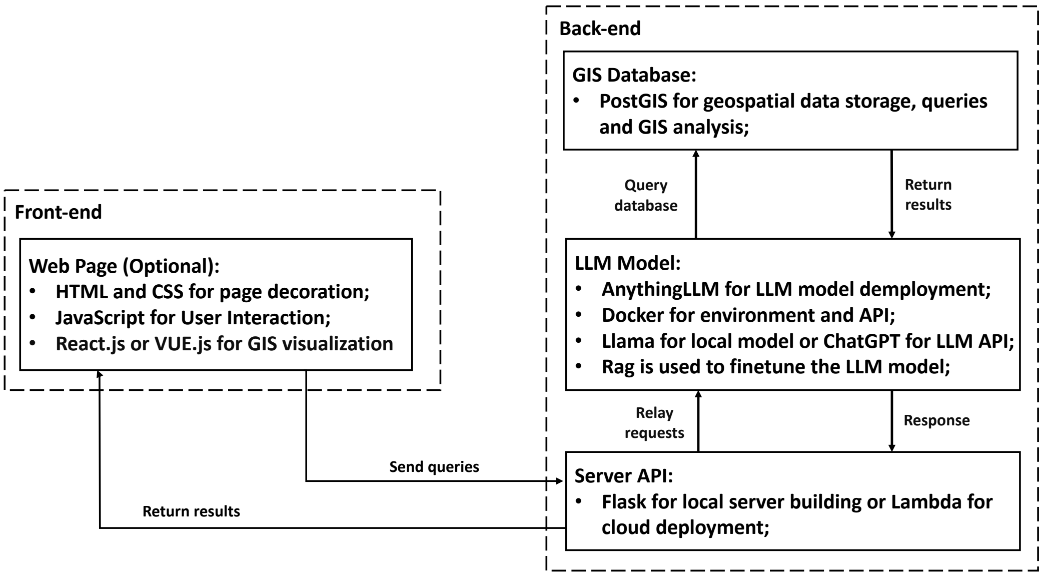
Back-end
GIS Database:
• PostGIS for geospatial data storage, queries and GIS analysis;
Query database
Return results
Front-end
Web Page (Optional):
• HTML and CSS for page decoration;
• JavaScript for User Interaction;
• React.js or VUE.js for GIS visualization
LLM Model:
• AnythingLLM for LLM model demployment;
• Docker for environment and API;
• Llama for local model or ChatGPT for LLM API;
• Rag is used to finetune the LLM model;
Relay requests
Response
Send queries
Return results
Server API:
• Flask for local server building or Lambda for cloud deployment;

Figure 1: System Overview

In this project, we will try to implement the local LLM model first and the LLM API (ChatGPT) second. Since this is the project for the single person, the front-end is the optional choice. The GIS database will be built using the OSM dataset in California. The server API will be built using Flask and deployed locally if time permits, we may try to deploy the whole system to the cloud using AWS or Google Cloud Platform.

3.2 LLM for Query Processing

In this project we will use the Llama model **?** as the LLM locally. Llama is a natural language processing model developed by the Stanford NLP group. Llama is a pre-trained transformer-based language model that can generate SQL queries. We will use the Llama model to parse the user queries and convert them into SQL statements. The schema of the GIS database will be used in the Rag to help optimize the SQL queries.

ChatGPT **?** is a pre-trained transformer-based language model that can generate natural language responses. We will use the ChatGPT API as the Back up plan to generate responses to the user queries.

3.3 Database & Storage

In this project, we will use the PostGIS database **?** to store and manage the geospatial data. PostGIS is a spatial database extender for PostgreSQL that adds support for geographic objects. The PostGIS database will be used to store and manage the geospatial data. The schema of the database will be designed based on the OSM dataset.

3.4 API & Integration

In this project, we will use Flask as the server API framework. Flask is a micro web framework written in Python. We will use Flask to provide a RESTful interface for the front-end and third-party applications to interact with the system. We will also provide a simple API endpoint to allow users to upload their own geospatial data to the system.

3.5 Deployment Plan

In this project, we will deploy the system locally for testing first. We will use Docker to containerize the system and the database and provide a simple deployment script to allow users to quickly start the system. If time permits, we can deploy the whole system to the cloud using AWS or Google Cloud Platform.

3.6 Visualization (Optional)

In this project, we will use the ArcGIS Pro software as the visualization tool to check the results of the spatial queries and the layers downloaded from the OSM dataset. ArcGIS Pro is a geospatial software application that provides a wide range of GIS tools and visualization techniques. We will use the ArcGIS Pro to visualize the results of the spatial queries. Besides, we will also provide a simple visualization tool to allow users to visualize their own geospatial data. We will use the VUE.js to create a simple map visualization tool.

4 Team Members & Roles

This is a single-person project. I will be responsible for the project design, data collection, document writing, project management, and the presentation of the project. I will also be responsible for the implementation of the system components and the integration of the components.

5 Timeline

Before the midterm submission, I plan to complete the following tasks:

1. Design the project proposal and write the project report.
2. Collect and clean the OSM dataset in California.
3. Build the GIS database using the OSM dataset.
4. Implement the local LLM model or the LLM API (ChatGPT).

After the midterm and before the final submission, I plan to complete the following tasks:

1. Implement the Flask API and integrate it with the front-end.
2. Test the system locally and provide a simple deployment script.
3. Write the final report and the presentation.
4. Deploy the system to the cloud if time permits.