Intelligent searching in geospatial Big Data

1st Wen-Hsin Chen
College of Engineering
University of Missouri - Columbia
Columbia, United States
mch84@missouri.edu

2nd Timothy Haithcoat
MU Institute for Data Science and Informatics (MUIDSI)
University of Missouri - Columbia
Columbia, United States
haithcoatt@missouri.edu

3rd Chi-Ren Shyu
MU Institute for Data Science and Informatics (MUIDSI)
University of Missouri - Columbia
Columbia, United States
shyuc@missouri.edu

***Abstract*—Access to spatially-integrated health, environmental, cultural, and social data can help researchers and practitioners gain a holistic perspective of a patient/subject through knowledge of their surrounding health access, environmental and or social risks. A platform enabling intelligent retrieval and recommendations of such user-based contextual queries on spatially integrated data does not currently exist, thus, this project aims to develop one. To engineer the platform's natural language search functionality, the proposed approach integrates Large Language Model (LLM) capabilities from Cohere's Command R+ model and utilizes the Cohere Python SDK. The search functionality operates by translating a natural language query into an equivalent Structured Query Language (SQL) statement, which is subsequently executed against the geospatial Big Data. Depending on the user's search intent, the retrieved SQL results can be returned directly or subjected to additional post-processing. In the latter case, the system performs two tasks: (1) Identify and recommend supplementary data attributes that may be of analytical interest yet were not explicitly requested by the user, and (2) Generate a concise summarization of only the information relevant to the initial search prompt and the recommended and supplementary data attributes. Evaluation will be conducted manually and using LLM-as-a-judge. Each output will be evaluated based on four criterions: compliance, correctness, relevance, and conciseness. This study develops a prototype, defining a general workflow for retrieval and recommendations of user-based contextual queries on spatially integrated data for researchers from various domains to evaluate query recommendation usefulness and suggest refinements to improve the quality of the results. While balancing large-scale retrieval systems is a challenge for performance speed and accuracy, future phases of this research will demonstrate the system's functionalities through domain-specific use-cases.**

***Index Terms*—Geospatial data, data retrieval, Agentic AI, Large Language Models, Natural Language Processing**

I. Introduction

Historically, scientific innovations have depended on reductionism to decompose complex systems down into atomized components and concentrated study on each of components in isolation. This technique advanced human understanding across various disciplines, from the structure of molecules to the inner-workings of biological processes. However, by detaching the individual from their ecosystem in research, scientists lose environmental context, the connection that binds data to the real-world system. Today, the amount of data that can be collected and consolidated is unparalleled to any other time throughout history, what is still lacking is the technology to relink these datapoints to their broader contexts. For instance, a medical practitioner or researcher may have access to electronic health records (EHR) and various other patient records to study a certain illness but they don't have immediate access to crucial environmental factors, resource availability, and geographic limitations that could just as well impact patient outcomes. This disconnect prevents these practitioners and researchers from gaining a holistic understanding of patients and making fully informed decisions. To address this gap, the present study proposes a framework to parse natural language search prompts and intelligently fetch the primary requested data alongside any critical contextual information—such as environmental factors, resource availability, and geographic constraints. This study enables researchers and practitioners to gain more contextual-based insights and consequently answers the following research questions:

- How can a technological system be architected to intelligently identify and retrieve relevant contextual data for a given primary research search prompt?
- Compared to isolated data analysis, how can retrieval of this contextual data advance research insights?

The central hypothesis of this work is that integrating a primary research query with a framework powered by Large Language Models (LLMs) and agentic AI will generate a more intelligent and context-aware retrieval workflow, thereby yielding more comprehensive geospatial insights than querying primary data in isolation.

II. Literature Review

With the rise of LLMs and Agentic AI, the retrieval of geospatial data is an area in crucial need of recontextualization and further exploration. Using these technological innovations, natural language queries for geospatial data retrieval can

be transformed into a more intelligent and comprehensive process. Several relevant machine learning, generative AI, and natural language processing methods have been proposed and implemented to address this task.

A. Retrieving Open Geospatial Consortium (OGC)-compliant geospatial datasets

Several relevant machine learning, generative AI, and natural language processing methods have been proposed and implemented to address this task. For instance, Elia Ferrari et al., from the Institute of Geomatics, FHNW University of Applied Sciences and Arts Northwestern Switzerland, proposes the proof of concept of a search engine designed to retrieve Open Geospatial Consortium (OGC)-compliant geospatial datasets in Switzerland [1]. The proposed research involves augmenting metadata through natural language processing and extracting keywords from abstracts of the data using a non-deep-learning graph method called Rapid Automatic Keyword Extraction (RAKE) then further keyword refinement by Sentence-BERT. Kendal tau distance was then used to evaluate the query result rankings. These methods are relevant to this research as they offer insight on how to augment natural language data and return more relevant query results.

B. Google's Geospatial Reasoning

In 2025, Google conducted research on Geospatial Reasoning through their AI model, Gemini, which currently can execute GIS operations and analysis on Google Earth. Geospatial Reasoning aims to extend Gemini's capabilities to answer complex geospatial natural language queries through Chain of Reasoning [2]. Although Google's specific methodologies have not been publicly disclosed, their research objective aligns closely with that of this research: To engineer a model that can consolidate public and private user datasets, drawing inference from this analysis, and efficiently generate truthful insights and useful data visualization.

III. Methodology

A. Materials

1) Dataset: The geospatial Big Data utilized in this study is only a small subset to a larger big database known as Geospatial Analytical Research Knowledgebase (GeoARK) [3]. GeoARK is a comprehensive database that integrates and flattens multiple GIS layers into one unified table. Spatial datapoints in GeoARK are interconnected by their locational attributes.

The subset of GeoARK employed in this research was pre-processed, cleaned, and subjected to analytical operations prior to use. The original data sources include publicly accessible repositories from the U.S. Geological Survey (USGS), USGS Watershed Boundary Dataset, Oak Ridge National Laboratory (ORNL), Homeland Infrastructure Foundation-Level Data (HIFLD), United States Environmental Protection Agency (EPA), and US Census Bureau Decennial Census (2010 and 2020). Additional attributes such as distance measures were derived through geoprocessing softwares such as Quantum Geographic Information System (QGIS) and ArcGIS (Environmental Systems Research Institute, ESRI) prior to use.

The dataset used in this study has 2747924 rows each with 115 columns. Each row in the dataset represents a 100-radius area in the U.S. and contains column attributes across domains such as soil and air quality, industrial activity, hospitals, demographics, and more.

2) LangGraph: LangGraph is an extension of the LangChain library. LangGraph is used as a tool to engineer stateful multiactor pipelines with LLMs. By using this tool, developers have the ability to add Directed Acyclic Graphs to their chains of computation. This permits more agentic behavior where LLMs could be called in a loop to determine the next step it should take base on the state that it has.

In LangGraph, the stateful graph passes around the same state which can be continuously altered by each node that it passes through in the graph. The nodes in the graph can represent a Python function or computational step that must be taken to complete some complex task. The edges define the direction to which the state will be passed to. The dotted edges as seen in Fig. 1 represents a conditional path that the state may or may not pass through based on the decisions of the source node.

LangGraph is licensed under the MIT License, making it free to use, modify, and distribute for both commercial and non-commercial applications. Alongside the abilities that it enables, this made LangGraph the ideal library for conducting this study.

3) The Cohere Command R+ Model: The Cohere Command R+ Model is a LLM model trained for multi-step agentic AIs, it allows a list of tools as inputs and then outputs a JSON list of sequential actions to execute based on the inputted list of tools. Compared to previous Command R+ versions, the August 2024 updated version used in this study utilizes has 50% higher throughput and 25% lower latencies while utilizing approximately the same amount of computational resources.

This study used the Cohere Python SDK, which is also released under the MIT License. It is an optimal model for the various LLM prompt calls necessary in the geospatial data search functionality because it is able to call upon SQL tools to help convert natural language prompts to SQL queries via the LangChain function $SQLDatabaseChain.from\_llm()$.

4) all-MiniLM-L6-v2: The all-MiniLM-L6-v2 is a sentence transformer model that encodes each sentence or short paragraph from a given text to a 384-dimensional dense vector space representation. It is derived from MiniLM-L6-H384-uncased and fine-tuned on one billion sentence pairs, where, given a random sentence from the pair, the model must predict its corresponding pair sentence from a set of randomly sampled sentences. This study employs the all-MiniLM-L6-v2 model because its intended usage for clustering and semantic similarity search tasks is suitable to the cosine similarity task that was performed in the geospatial search functionality workflow. A Python dictionary, keyed by column names that have been abbreviated and valued by the column name de-

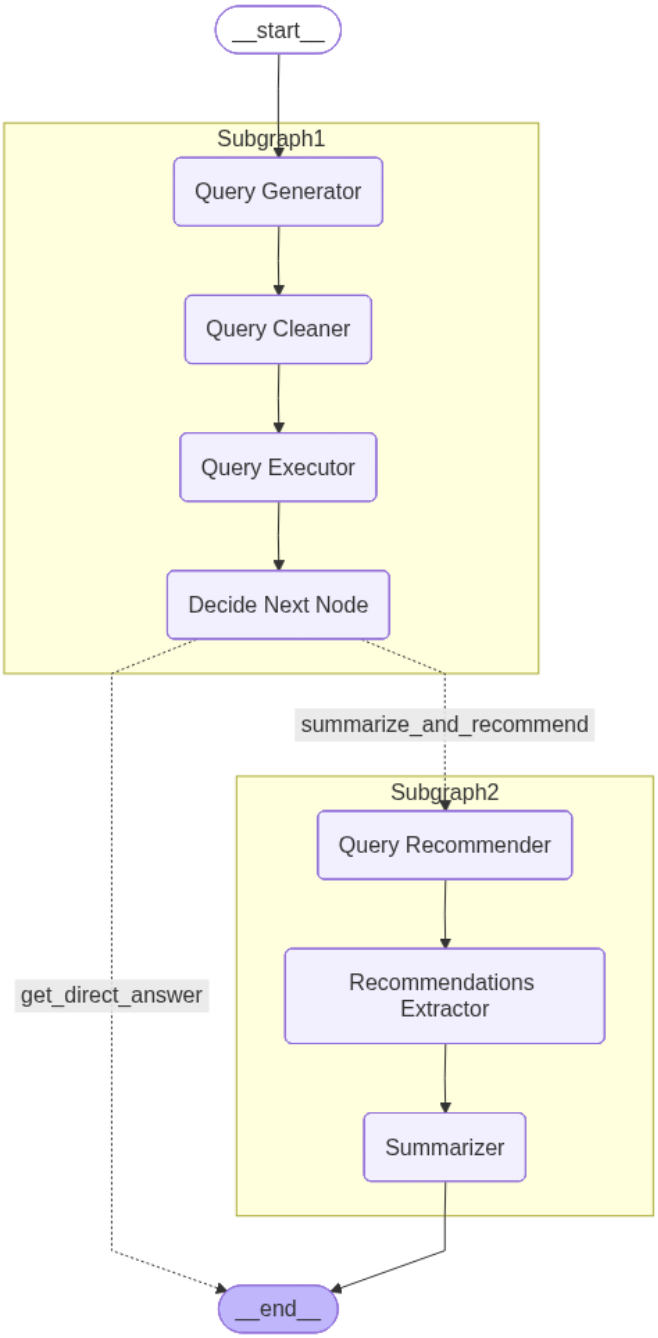

Fig. 1. LangGraph graph workflow

scriptions, was encoded beforehand using all-MiniLM-L6-v2, this encoding can be retrieved during execution or production. This reduces the computational time and resources needed for repeated and unnecessary encoding.

B. Method

The LangGraph graph was split into two subgraphs, labeled Subgraph1 and Subgraph2 in Fig. 1. In this paper, they are respectively referred to as the initial processing steps and the further processing steps. The initial processing steps—Subgraph1–contain four nodes were executed in the following sequential order:

1) Query Generator node
2) Query Cleaner node
3) Query Executor node
4) Decide Next Node node

The further processing steps—Subgraph2–contain three nodes were executed in the following sequential order:

5) Query Recommender node
6) Recommendations Extractor node
7) Summarizer node

In addition to passing the user's raw natural language into the workflow, an additional system message was added. The system message instructed the LLM to generate SQL queries that fetch entire row of any retrieved data. This ensured that the retrieved SQL results can be as consistent as possible for further processing down the workflow.

1) Query Generator node: The Query Generator node generated an SQL query based on the user's provided natural language prompt. The LLM used to generate this conversion was the Cohere Command R+ model and it was forced to return an SQL output by setting $return\_sql$ as $True$ in the function call $SQLDatabaseChain.from\_llm()$.

2) Query Cleaner node: The Query Cleaner node sanitizes the raw output obtained from the previous Query Generator node. This prevents the query-executing SQLlite engine from raising query formatting errors. The raw output from the previous node was usually in markdown code syntax, to convert it into an SQL query it required stripping off the trailing 'S's, wrapping backticks, and potentially extraneous whitespaces. This node could have also been been revised to have a LLM evaluate and clean the previous node's output but since the outputs almost never included other syntactical errors, this cleaning step was hard-coded to save computational resources and improve generation speed.

3) Query Executor node: The Query Executor node used the database engine to execute the cleaned SQL query from the previous Query Cleaner node. The fetched SQL result would return as a Python list so it was converted into a string before being sent to the next step in the workflow.

4) Decide Next Node node: The Decide Next Node node embodied a simple if statement that determined if the user's original input contained the decision to continue to further processing or if it desired a direct answer. The default was continuing to further processing and taking the $summarize\_and\_recommend$ path. However, the user could also opt to directly receive the output of the previous Query Extractor node and skip the further processing workflow path. If the user choose to further process the data, the workflow continued...

5) Query Recommender node: The Query Recommender node entered the original user's search prompt from the start of the LangGraph as well as into the Cohere Command R+ model. The LLM was instructed to use both of these texts to recommend additional points of exploration. For instance,

in prompt 3: LLM may note the words “Child care” in the prompt and recommend topics related to schools or daycare facilities as further points of exploration.

6) Recommendations Extractor node: The Recommendations Extractor node compared the cosine similarities between corpus embeddings and recommendations text generated from the previous Query Recommender node. As stated previously, the all-MiniLM-L6-v2 sentence transformer model has already created the corpus embeddings from a list of tuples with abbreviated column names and their brief description values. Thus, the only text encoding computation required in this node was for the generated recommendation text to be encoded which can be done using the same method as the corpus embeddings. Once the top k number of encodings with the highest cosine similarity scores have been found, they are matched back to their actual values from past Query Executor node's output. The top k number of matches was formatted and concatenated into a string that will include the name of each matched column, the brief description of the column, the cosine similarity score and the value in the column. This type of extraction process was necessary to filter out irrelevant columns and their data which may number close to a hundred and narrow the output down into only the top-k number of columns that are most relevant to the user's querying objectives.

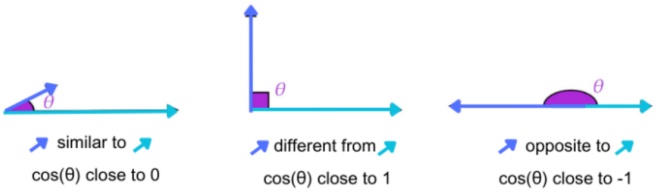

Fig. 2. Cosine similarity comparisons on 2-dimensional embeddings

7) Summarizer node: The Summarizer node receives the output from the previous Recommendations Extractor node and is instructed to summarize the features and values by the human prompt. The system message assigns the role of “data summarization assistant” to the LLM. The messages are entered into the Cohere Command R+ model. The natural language text summary that the model outputs is returned to the user as the final output.

IV. Results

7 various natural language prompts were tested through the workflow. The nodes in the workflow will be evaluated based on either their compliance, correctness, relevance, or conciseness.

To maintain impartiality, these results were evaluated using LLM-as-a-judge for compliance, relevance, and conciseness criterions. The DeepSeek-V3.2-Exp model was employed as an evaluator for the task. The model was provided the same criterion and description as listed below then given the responding data that each criterion evaluates.

- **Compliance** evaluates how closely-aligned the generated SQL query is to what the user's natural language search prompt was asking for. This criteria disregards the direct answer and the additional recommendations and summary generated from further processing, focusing only on the sanitized SQL query generated at the second node in the initial processing component. Compliance is rated from a scale of 0-10 with 0 meaning non-compliant and 10 meaning fully-compliant.
- **Correctness** evaluates how closely-aligned the output is to the actual truth. This criteria disregards the additional recommendations and summary generated from further processing, focusing only on the direct answer output from the initial processing component. Correctness is rated from a scale of 0-10 with 0 meaning completely incorrect and 10 meaning completely correct.
- **Relevance** evaluates how closely-aligned the recommendations are from the user's natural language query. Relevance is rated from a scale of 0-10 with 0 meaning completely irrelevant and 10 meaning completely relevant.
- **Conciseness** evaluates how void of irrelevant data is the summarized output from further processing step. Conciseness is rated from a scale of 0-10 with 0 meaning not concise and 10 meaning most concise.

Table I evaluated the workflow when the Cohere Command R+ model temperature was set to 1.0 and the top_k values for cosine similarity scores was set to 5. The same prompts with the same hyperparameters were ran for 3 generations or trials. The scores listed in Table I is the averaged score across the 3 generations.

Subtable 1

Prompt	Compliance	Correctness
”Which object is closest to VHA?”	8.7	10
”Return an object with the highest number of Oil Refineries around it.”	8.7	10
”Show the objects near a Nonferrous Metal Mine.”	6	10
”Child care facility in Columbia”	6.7	3.3
”Find public health facilities within 50 km of Columbia, Missouri.”	3	0
”Which region shows the lowest average distance to hospitals?”	7	0
”List attributes for the 2 objects near Miscellaneous Industry.”	4.3	5.3

Subtable 2

Prompt	Relevance	Conciseness
”Which object is closest to VHA?”	8	10
”Return an object with the highest number of Oil Refineries around it.”	10	10
”Show the objects near a Nonferrous Metal Mine.”	6	7
”Child care facility in Columbia”	8	10
”Find public health facilities within 50 km of Columbia, Missouri.”	4	10
”Which region shows the lowest average distance to hospitals?”	10	10
”List attributes for the 2 objects near Miscellaneous Industry.”	4.7	10

TABLE I
Generation/Trial 0, 1, 2. Temperature = 1.0. Top_k = 5.

Table II evaluated the workflow when the Cohere Command R+ model temperature was set to 0.5 and the top_k values for cosine similarity scores was set to 5.

Subtable 1

Prompt	Compliance	Correctness
"Which object is closest to VHA?"	8	10
"Return an object with the highest number of Oil Refineries around it."	8	10
"Show the objects near a Nonferrous Metal Mine."	6	10
"Child care facility in Columbia"	1	10
"Find public health facilities within 50 km of Columbia, Missouri."	4	0
"Which region shows the lowest average distance to hospitals?"	9	0
"List attributes for the 2 objects near Miscellaneous Industry."	3	0

Subtable 2

Prompt	Relevance	Conciseness
"Which object is closest to VHA?"	10	10
"Return an object with the highest number of Oil Refineries around it."	8	10
"Show the objects near a Nonferrous Metal Mine."	10	10
"Child care facility in Columbia"	8	10
"Find public health facilities within 50 km of Columbia, Missouri."	8	10
"Which region shows the lowest average distance to hospitals?"	10	10
"List attributes for the 2 objects near Miscellaneous Industry."	2	10

TABLE II
GENERATION/TRIAL 0. TEMPERATURE = 0.5. TOP_K = 5.

Table III evaluated the workflow when the top_k values for cosine similarity scores was set to 3 and the temperature of the Cohere Command R+ model was set to 1.0.

Subtable 1

Prompt	Compliance	Correctness
"Which object is closest to VHA?"	10	10
"Return an object with the highest number of Oil Refineries around it."	8	10
"Show the objects near a Nonferrous Metal Mine."	6	10
"Child care facility in Columbia"	8	10
"Find public health facilities within 50 km of Columbia, Missouri."	9	0
"Which region shows the lowest average distance to hospitals?"	2	0
"List attributes for the 2 objects near Miscellaneous Industry."	2	10

Subtable 2

Prompt	Relevance	Conciseness
"Which object is closest to VHA?"	10	10
"Return an object with the highest number of Oil Refineries around it."	10	10
"Show the objects near a Nonferrous Metal Mine."	10	10
"Child care facility in Columbia"	9	10
"Find public health facilities within 50 km of Columbia, Missouri."	3	10
"Which region shows the lowest average distance to hospitals?"	10	10
"List attributes for the 2 objects near Miscellaneous Industry."	10	10

TABLE III
GENERATION/TRIAL 0. TEMPERATURE = 1.0. TOP_K = 3.

V. DISCUSSION

The results indicate that while easily-interpretable queries such as the first few prompts yield good overall total scores, as the workflow attempted to answer more complex or vague questions, it often made errors that ended up generating non-compliant or incorrect outputs.

One aspect that simplified the evaluation and coding process was the transparency of the steps taken by the workflow. At each node, an output can be returned and printed to the screen to display if there are any discrepancies with query generation, additional recommendation, and or final summarization.

Future work will aim to create a more robust workflow that can handle various natural language search prompt inputs without generating incorrect data. The context window size for the Cohere R+ Command model is 128,000 tokens therefore, improvements can be made with the system prompt to include more detailed instructions on how to handle edge cases.

VI. CONCLUSIONS

This study developed a prototype workflow for a geospatial search functionality using agentic AI and LLMs to divide the search task into two core components: The initial processing component and the further processing component. In the initial processing component, a user inputs a natural language prompt, the prompt is then converted into a SQL query using the Cohere Command R+ Model, the generated query is sanitized, then ran on the SQL database to output the retrieved datapoint(s). If the user initially elects for the search functionality to undergo further processing, their original natural language prompt is passed along with the retrieved datapoint to the further processing component. The further processing component will instruct the Cohere Command R+ Model to generate recommendations based on the user's original natural language search prompt. The recommendation text is then encoded and compared against the corpus embeddings of the dataset's column names and their descriptions, the top k number of encodings with the highest cosine similarity score is mapped to the their fetched values generated at the end of the initial processing component, the final result is the desired answer to the query as well as recommended data attribute values of the same row for user's further exploration.

Although some prompts presented challenges for the workflow in fetching data. The results of the project revealed the power of the LLMs and the agentic AI workflow for explainable AI. At every step of the workflow, results were documented and could later be used for debugging or prompt refinement. For subsequent studies, there remains much work to be done to enable the workflow to study the patterns across the dataset and reveal any geospatial hotspots that may present more insight to researching users.

REFERENCES

[1] D. Schottlander and T. Shekel, "Geospatial reasoning: Unlocking insights with Generative AI and Multiple Foundation models," Google Research, 08-Apr-2025. [Online]. Available: https://research.google/blog/geospatial-reasoning-unlocking-insights-with-generative-ai-and-multiple-foundation-models/. [Accessed: 20-Sep-2025]

[2] E. Ferrari, F. Striewski, F. Tiefenbacher, P. Bereuter, D. Oesch, and P. Di Donato, "Search engine for Open Geospatial Consortium Web Services improving discoverability through natural language processing-based processing and ranking," ISPRS International Journal of Geo-Information, vol. 13, no. 4, p. 128, Apr. 2024. doi:10.3390/ijgi13040128

[3] T. Haithcoat, D. Liu, T. Young, and C.-R. Shyu, "Investigating health context using a spatial data analytical tool: Development of a geospatial Big Data Ecosystem," JMIR Medical Informatics, vol. 10, no. 4, Apr. 2022.