International Journal of Digital Earth

ISSN: 1753-8947 (Print) 1753-8955 (Online) Journal homepage: www.tandfonline.com/journals/tjde20

An autonomous GIS agent framework for geospatial data retrieval

Huan Ning, Zhenlong Li, Temitope Akinboyewa & M. Naser Lessani

To cite this article: Huan Ning, Zhenlong Li, Temitope Akinboyewa & M. Naser Lessani (2025) An autonomous GIS agent framework for geospatial data retrieval, International Journal of Digital Earth, 18:1, 2458688, DOI: 10.1080/17538947.2025.2458688

To link to this article: https://doi.org/10.1080/17538947.2025.2458688

Published online: 09 Feb 2025.

INTERNATIONAL JOURNAL OF DIGITAL EARTH
2025, VOL. 18, NO. 1, 2458688
https://doi.org/10.1080/17538947.2025.2458688

An autonomous GIS agent framework for geospatial data retrieval

Huan Ning, Zhenlong Li, Temitope Akinboyewa and M. Naser Lessani

Geoinformation and Big Data Research Laboratory, Department of Geography, The Pennsylvania State University, University Park, USA

ABSTRACT

Powered by the emerging large language models (LLMs), autonomous geographic information system (GIS) agents can perform spatial analyses and cartographic tasks. However, a research gap exists in enabling these agents to autonomously discover and retrieve the necessary data for spatial analysis. This study proposes an autonomous GIS agent framework capable of retrieving required geospatial data by generating, executing, and debugging programs. The framework, with an LLM-driven decision core, selects data sources from a predefined list and fetches data using source-specific handbooks that document metadata and data retrieval details. Designed in a plug-and-play style, the framework allows human users or automated data crawlers to add new sources by creating additional handbooks. A prototype agent based on the framework is developed and released as a QGIS plugin and a Python program. Experiment results demonstrate its capability of retrieving data from various sources, including OpenStreetMap, administrative boundaries and demographic data from the U.S. Census Bureau, satellite basemaps from ESRI World Imagery, global digital elevation model (DEM) from OpenTopography.org, weather data from a commercial provider, and the COVID-19 case data from the NYTimes GitHub. This study is among the first attempts to develop an autonomous GIS agent for geospatial data retrieval.

ARTICLE HISTORY
Received 28 August 2024
Accepted 21 January 2025

KEYWORDS
Autonomous GIS; geospatial data retrieval; large language models; generative AI; GIS agent; AI assistant

1. Introduction

In recent years, large language models (LLMs) have drawn tremendous attention from researchers. These models are trained on massive text corpora and are able to reason and respond to questions (prompts) in natural language. Serving as reasoning cores, LLMs are able to make decisions and assist analysis for various applications, such as robotics (Sun et al. 2024) and clinical diagnosis (Hong et al. 2024). LLM also has the potential to power geographic information systems (GIS), enabling GIS tools to perform spatial analysis tasks autonomously. Researchers in the GIS community have attempted to implement such autonomous GIS agents, aiming to automate spatial analysis (Akinboyewa, Li, et al. 2024; Li and Ning 2023; Zhang et al. 2024a), cartography (Tao and Xu 2023; Zhang et al. 2024b), and disaster management (Akinboyewa, Ning, et al. 2024;

CONTACT Zhenlong Li zhenlong@psu.edu Geoinformation and Big Data Research Laboratory, Department of Geography, The Pennsylvania State University, University Park, USA

Hao et al. 2024). The main purpose of those autonomous agents is to promote productivity and democratize GIS technologies by reducing human intervention.

For all spatial analyses, GIS agents need to access essential geospatial datasets, such as study area boundaries, points of interest (POIs), imagery, and attributes of geographic entities. Li and Ning (2023) highlighted that online geospatial data discovery and filtering are necessary capabilities of autonomous GIS. Their experiments showed that LLM is able to obtain data from REST (Representational State Transfer) APIs with brief documentation about endpoints, parameters, and format of the returned data. In their study, however, most experimental datasets were provided by local file paths or URLs. Their work on autonomous GIS did not propose solutions or mechanisms for geospatial data retrieval.

Geospatial data can be sourced from a variety of online and local repositories. For example, in various research contexts, OpenStreetMap, data portals from governmental agencies, and satellite imagery platforms are widely used as open-access data sources. Automatically retrieving data from these extensive data sources can enhance GIS agents' autonomy, potentially enabling fully autonomous geospatial research and knowledge discovery. Tools like GeoGPT (Zhang et al. 2024a) and GeoQAMap (Feng et al. 2023) use LLMs for data collection and query conversion, showing potential for various applications. Other tools, such as GeoForge (Ageospatial 2024), integrate GPT models for geospatial data analysis but remain limited in flexibility and data source support. While these studies highlight the feasibility of LLM-powered GIS agents for geospatial data queries, the lack of universal mechanisms, extensible frameworks, or implementations remains a significant gap in the literature.

Autonomous geospatial data retrieval faces several critical challenges, including (1) discovering relevant data from various online sources. While search engines like Google can retrieve data based on keywords, selecting and assessing appropriate geospatial data requires comprehensive strategies. This process is both time-consuming and intelligence-intensive; (2) selecting the appropriate data sources by interpreting data requests made in natural language or specific requirements, which may not always clearly designate the data source. The autonomous agent needs to understand such requests and determine which data source should be used; and (3) fetching data from specific sources involves varying technical methods. Some data sources provide API (Application Programming Interface) access, downloading URLs, and sometimes require API keys or passwords. Autonomous agents need to be flexible, knowledgeable, and 'smart' to handle the parameters in APIs or URLs correctly.

To address this gap, we designed an autonomous GIS agent framework, named LLM-Find, for geospatial data retrieval, aiming to address the second and third challenges. The challenge of online data source discovery will be explored in future research as it requires intensive research into web search and analysis. The framework retrieves requested data by generating, debugging, and executing programs based on natural language queries. It is expandable and flexible, utilizing handbooks for each data source that provide necessary metadata and technical information for data fetching. To demonstrate the concept and feasibility of the framework, we developed a prototype agent including six data sources: OpenStreetMap, US Census data, weather data, ESRI (Environmental Systems Research Institute, Inc.) satellite images, COVID-19 data, and worldwide digital elevation models (DEMs). New data sources can be added to the agent by adding new handbooks in a plug-and-play manner. For over 70 data request tests, the agent obtained the correct data with a high chance of 80%–90%, demonstrating its potential for automating geospatial data retrieval for autonomous GIS.

We implemented the prototype agent in two formats: a QGIS plugin and an interactive Python program in Jupyter Notebook. The QGIS plugin allows GIS users to download geospatial data without coding and use the downloaded data directly in a GIS environment. Meanwhile, developers can use the Python program to download data with greater flexibility and customization, such as automating geoprocessing tasks or making the agent collaborate with other programs or agents.

2. Related work

LLMs have drawn remarkable attention from various domains due to their excellent ability to understand natural language. Researchers have explored various LLM applications in GIS (Wang et al. 2024). In the data retrieval domain, scholars mostly used LLM as an information extractor to retrieve geospatial information of interest or as a decision core to create data retrieval workflows.

2.1. Geospatial information retrieval based on LLMs

Li and Zhang's (2023) work on MeataQA aims to enhance the spatial data search for human users. Users can ask MetaQA through natural languages instead of keywords and drop-down menus. This system provides users with requested datasets from a complex metadata table. There are many other attempts to use LLM to extract target information from text or images. For example, Crooks and Chen (2024) suggested the new frontier of urban information extraction based on LLMs, such as deriving descriptions and locations from street images. Similarly, Hu et al. (2023) demonstrated the usage of LLMs in extracting the location descriptions from disaster-related social media messages. Extracting Point of Interests (POIs) from text data is a common procedure in spatial analysis. Kim and Lee's (2024) study on their LLM-based tool, POI GPT, demonstrated its effectiveness in extracting place names, addresses, classifications, and geographic coordinates, with an accuracy of 78.51% and precision of 53.19%.

Mohanad Diab (2024) reported the geographic information types that can be retrieved from LLM, including landmark locations, population, and remote sensing images via Google Earth Engine API. LLMs also provide the possibility of conveying domain knowledge to users in a conversational manner. Using retrieval-augmented generation techniques (Zhao et al. 2024), Mosser et al. (2024) created a chatbot to retrieve geological knowledge databases and generate answers for user-posted questions. This process involved LLMs generating code and SQL queries to extract information from existing databases. LLM-based information retrieval also enables semantic and context search rather than only relying on traditional keyword matching. Menezes (2023), for example, showed the effectiveness of combining traditional keyword matching with semantic search for extracting unstructured data in the petroleum industry. Another study (Grant et al. 2024) on subsurface modeling also demonstrated LLMs' ability for information extraction by developing a copilot to extract target information from unstructured text data for a domain model.

2.2. Geospatial data retrieval based on LLMs

To retrieve geospatial data based on LLMs, researchers made their attempt on both pre-trained models, such as OpenAI GPT, and trained customized models (Ma et al. 2023), like Llamma models. In the study by Mansourian and Oucheikh (2024), Llama 2 model (Touvron et al. 2023) was fine-tuned to generate Python script for QGIS to access external datasets such as OpenStreetMap. In a cartography test (Tao and Xu 2023), GPT showed the ability to obtain online available datasets, such as U.S. jurisdiction boundaries, to execute mapping tasks without provided datasets. Zhang et al. (2024a) proposed an autonomous GIS framework, GeoGPT, to collect, process, and analyze geospatial data. GeoGPT is able to download POIs and road networks from OpenStreetMap and remote sensing images from Geospatial Data Cloud. However, this work did not provide details on the implementation of the data fetching module and its expanding mechanism. The work by Feng et al. (2023) on GeoQAMap exhibited the idea of querying data from an online Wikidata and Query Service data source. GeoQAMap first adopted an LLM (GPT-3.5, OpenAI 2024) to convert the spatial questions on geo-entities to SPARQL queries, and then retrieved related geo-entities from Wikidata. The authors suggested that GeoQAMap could potentially be applied to other data sources, such as OpenStreetMap and CityGML, but did not provide further investigation or a general framework for data retrieval.

Similar to GeoQAMap, Staniek et al. (2023) also investigated the feasibility of data fetching from OpenStreetMap by applying fine-tuned T5 models (Raffel et al. 2020) and GPT-4 (OpenAI 2024) to convert questions in natural language to queries in OverpassQL, a query language for OpenStreetMap data. GeoForge (Ageospatial 2024) is a web application powered by GPT that can request, process, analyze, and display satellite imagery, building footprints, and administrative boundaries. Its goal is to democratize geospatial information and analysis. However, this closed-sourced application is still in development, with limited data sources and no support for customized and local data sources. Geode (Gupta et al. 2024) is a geospatial question-answering agent with precise spatial–temporal data retrieval. It utilized APIs to retrieve information of geographic coordinate, boundary, weather, elevation, air quality, and raster. The backed LLM used the retrieved data to answer users' inquiries about locations. However, this work is not focused on data retrieval and downloading, and the extensibility was also not considered.

NASA and Microsoft recently launched the Earth Copilot project to simplify access to complex spatial data so that users without geospatial expertise can use plain language to explore Earth observation data (Bryson 2024). Additionally, the LLM providers, including Claude (Anthropic 2024) and OpenAI (Ghaffary and Metz 2024), are developing AI agents to operate computers by seeing the screen like humans. If these attempts become applicable in the following years, they have great potential to be used for autonomous geospatial data retrieval. Our study adopted the LLM as the reasoner to determine which data source satisfies the users' data requests and then generate data retrieving code to fetch requested data. Per our knowledge, the proposed LLM-Find is the first universal framework for geospatial data retrieval, supporting customized data sources.

3. Methodology

Typically, GIS analysts find and download geospatial data in three steps: (1) collecting potential data sources and their metadata, such as extent, resolution, date, and attributes. This procedure may involve using search engines or consulting colleagues; (2) determining which data source is most suitable for the analysis; (3) fetching the data based on the technical requirements set by the data providers, such as accessing URLs, using registered accounts, understanding data structures, and handling data chunking. In other words, human analysts are required to follow technical handbooks to retrieve the data.

Correspondingly, a geospatial data retrieval agent is expected to conduct these three steps autonomously. Since a GIS analyst is often specified in one or several domains, in this study, we began with a manually created GIS data source list and handbooks and focused on the automation of the latter two steps: determining the applicable data source from an existing list and fetching data from the selected source. This section introduces the proposed framework, LLM-Find, and explains its implementation details through a proof-of-concept prototype.

3.1. The framework

The framework comprises two major components: a data source index and a handbook inventory (Figure 1). The data source index provides brief introductions to various data sources, enabling LLMs to select the most appropriate source based on the data request. The handbook inventory contains technical handbooks detailing how to fetch data from each source, such as accessing entries and data formats. The aim is to provide sufficient information to reduce uncertainty for the agents. Additionally, two supplementary components, *authentication* and *codebase*, are used to manage source authorization and provide example code. The framework is designed to be straightforward and extendable to accommodate various data sources across different application scenarios. This design allows both humans and other autonomous agents to update or add new data sources in a plug-and-play manner.

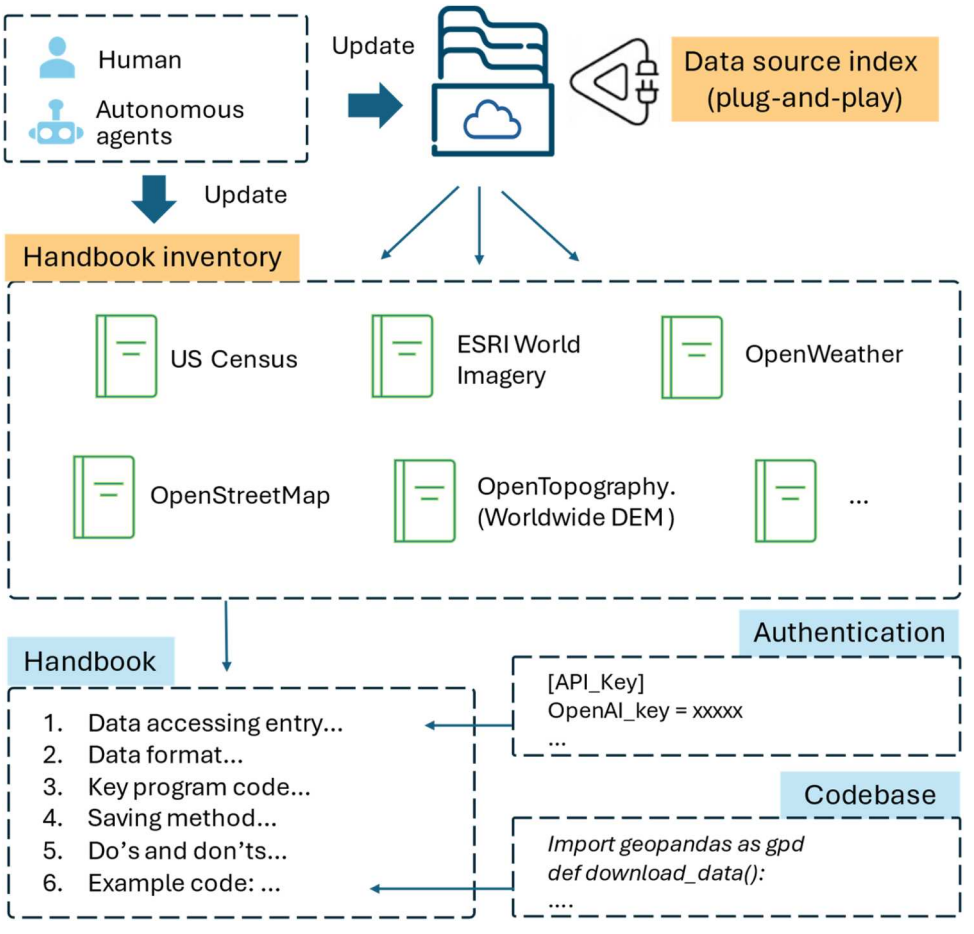


Figure 1. The architecture of the autonomous geospatial data retrieval framework.

3.1.1. Data source index

The data source index includes the source names and concise descriptions, informing the agent about the data sources it can use. A sample of the data source index is shown in Box 1. Note that the data source index should be brief and informative since it needs to be embedded in the prompt. Because the descriptive names may be lengthy and unsuitable for use as data source identifiers (IDs), we defined short aliases for them to serve as IDs.

Box 1. An example of the data source index.

1. OpenStreetMap. You can download the administrative boundaries, street networks, points of interest (POIs) from OpenStreetMap.
2. US Census Bureau boundary. It provides the US administrative boundaries (nation, state, county, tract, and block group level, as well as metropolitan statistical areas.
3. US Census Bureau demography. It provides the demographic and socio-economic data, such as population, gender, income, education and race.
4. OpenWeather data. It provides historical, current, and forecast weather data. The historical data can be back to 2023-08. API limits: [Hourly forecast: 4 days, Daily forecast: 16 days, 3 h forecast: 5 days]
5. ESRI World Imagery (for Export). It is a web map service, providing satellite image tiles. You can download tiles and mosaic them into a large image.
6. US COVID-19 data by New York Times. Cumulative counts of COVID-19 cases and deaths in the United States, at the state and county level, over time from 2020-01-21 to 2023-03-23.
7. OpenTopography. You can download global digital elevation model (DEM) data using API; the resolution ranges from 15m to 1000m, such as SRTM GL3 (global 90m), and GL1 (global 30m). The DEM source list from this API contains: SRTMGL3, SRTMGL1, SRTMGL1_E, AW3D30, AW3D30, SRTM15Plus, NASADEM, COP30, COP30, EU_DTM, GEDI_L3, GEBCOIceTopo, GEBCOSubIceTopo.

3.1.2. Handbook inventory

Each handbook in the inventory provides the necessary technical information for LLMs to retrieve data from its associated source. Although LLMs have been trained on massive texts, they still lack the detailed knowledge required to retrieve GIS data from various sources. Handbooks essentially fill the knowledge gaps in LLMs for particular geospatial data sources. They might cover data locations, entry endpoints, formats, and attributes. For sources requiring authentication (e.g. API keys) or specialized code to download and process data, the handbook provides this information. Since the framework generates programs to retrieve data, we found that providing it with a verified program template can significantly increase its code generation accuracy and reliability. These template programs are integrated into prompts and managed within a supplementary component, *codebase*, to ensure better maintenance and organization of these templates. In addition, the framework includes a dedicated component to manage authentication for security concerns. Each handbook serves a single data source and is maintained individually. The framework allows to add, update, and remove data sources by modifying the data source index and handbook inventory, making it plug-and-play. While humans can manually create and maintain these data sources, autonomous agents can also perform this task by crawling and assessing data sources and automatically generating handbooks with minimal human intervention.

Handbooks vary among data sources. For instance, OpenStreetMap provides the Overpass Query Language for API data retrieval (OpenStreetMap 2024), while the U.S. Census Bureau offers download links for administrative boundaries (US Census Bureau 2024). There may also be multiple methods or sources to retrieve the data. For example, GIS analysts can obtain OpenStreetMap data using the official website's export function, the Overpass API, or Python packages like OSMnx, each with its advantages and disadvantages. Moreover, both OpenStreetMap and the U.S. Census Bureau provide U.S. administrative boundaries, so determining the better fit for a specific spatial analysis is crucial. Therefore, the agent needs to be provided with the necessary information via handbooks to ensure reliable data fetching.

In this study, we manually created handbooks based on intensive experiments to ensure agents can correctly understand the data request. Box 2 lists a few guidelines in the handbook for fetching OpenStreetMap data. Users can create and load a handbook for a specific data source by following the pre-defined template, which includes three parts: *data_source_name*, *brief_desciption*, *handbook*, and the optional *code_example*. For instance, if a user wants to add a data source of USGS (United States Geological Survey) Groundwater Levels Service (waterservices.usgs.gov) to the LLM-Find agent, they need to follow the template shown in Box 2. It is an iterative procedure to develop a valid handbook per our experience: the user needs to test and verify their customized handbooks using typical cases until it incorporates necessary data access information for the LLM and the success rate is acceptable. A quick start is copying the existing data source document as the *handbook*, and then refining it by testing various data requests.

Box 2. Parts of the handbook for retrieving OpenStreetMap data.

data_source_name = "OpenStreetMap"
brief_description = "You can download the administrative boundaries, street networks, points of interest (POIs) from OpenStreetMap."
Handbook = "1. In the requested area is given in an English name, you need to use '['name:en'='XX']' to filter the place in Overpass queries, otherwise you will get empty results. The 'name' tag in OpenStreetMap usually is in the location language.
2. If you need to download the administrative boundary of a place from OpenStreetMap, please use a Python package named 'OSMnx' by this code line: 'ox.geocode_to_gdf (query, which_result = None, by_osmid = False, buffer_dist = None)'. This method is fast
3. If you need to download POIs, you may use the Overpass API, which is faster than OSMnx library. Code example is: 'area['SO3166-2' = 'US-PA']->.searchArea;(nwr[amenity = 'hospital'](area.searchArea););out center;'
4. If you need to download polylines, you may use the Overpass API, which is faster than OSMnx library.
5. If you need to use a boundary to filter feature in GeoPandas, this is the code:'gpd.sjoin(gdf, boundary, how = 'inner', op = 'within')' … "
code_example = "#This is a program for your reference, note that you can improve it:
import geopandas as gpd

```
def download_data():
    overpass_url = 'https://overpass-api.de/api/interpreter'
    overpass_query = … "
```

3.2. The proof-of-concept agent

Based on the framework, we developed a proof-of-concept prototype agent that utilizes GPT-4o as the decision core to select the data source and generate Python programs to retrieve geospatial data. Figure 2 demonstrates the workflow of the agent. First, the agent receives the data request described in natural language and sends it along with the data source index to LLM to select the most appropriate source. The data request is straightforward, such as 'Download the 2021 population by race for each county in the US. Upon receiving the LLM's reply, the agent extracts the selected data source and retrieves the handbook associated with the selected source, which includes necessary authentications and template programs.

Appendix 1 shows an example prompt for selecting the data source. Users may specify which data source they need in the request, but it is often unnecessary since the backed LLM will select the most appropriate data source. However, in some scenarios, users should provide detailed requirements to ensure the results meet the expectations. For example, the ESRI World Imagery level and the resolution or provider of the DEM should given to the agent because agents do not have the context information to determine raster resolution.

With the data source identified and the handbook retrieved, the agent will then ask the LLM to generate Python code to fetch data. The handbook that contains the technical requirements will be sent with the code generation request. The agent employs a restricted program structure to ensure the code runs in a Python environment. For example, the LLM is instructed to generate the program using a pre-defined structure, such as placing all code within a 'download_data()' function rather than arbitrary names or the '__main__' function. The agent incorporated such information and requirements into the prompt to get the data downloading code (a prompt example can be found in Appendix 2). Finally, the agent executes the generated code to fetch the data. In our practice, the template program in the handbook plays an important role to help LLM generate the correct code to retrieve data, increasing the agent's success rate.

Recognizing that the LLM's spatial programming ability is limited due to insufficient training data and the complexity of the data-fetching programs, it is rare for LLMs (and humans) to write a bug-free program on the first attempt. Code debugging is a common and necessary step in real-world programming. Therefore, we implemented a self-debug module for the agent to

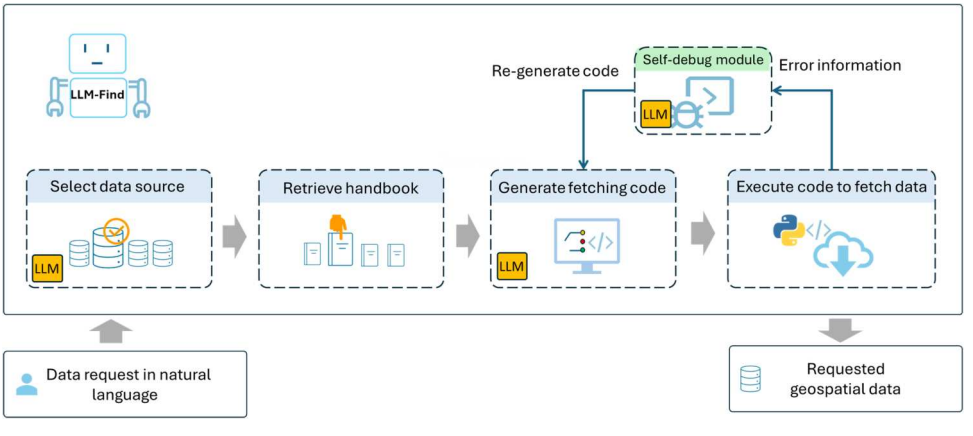


Figure 2. Workflow of the prototype agent based on the proposed framework.

correct the buggy code according to the error information. The self-debug module sends the necessary information to LLM and asks it to correct the code. The sent information includes the data request, handbook, generated program, occurred error, and debugging requirements. The re-generated code will be executed again to fetch the data. This execution-debugging process often runs multiple times until the generated program is error-free or the iteration times exceed the predefined limitation (which is set to 10 times in the prototype agent).

We developed the agent in two forms: a QGIS plugin (*Autonomous GIS – GeoData Retrieve Agent*) and an interactive Python program in Jupyter Notebook. QGIS is a free, open-source, and cross-platform GIS software, supporting geospatial data visualization and analysis. Around 2 million users install QGIS worldwide (Gispo 2021). The QGIS plugin can help these users to retrieve and load data using one simple sentence in natural language, aiming to improve their work efficiency. The interactive Python program allows GIS programmers to download data with greater flexibility and customization, such as automating their geoprocessing tasks or making the agent collaborate with other programs or agents. Figure 3 shows the QGIS plugin with downloaded data. The plugin is docked on the right side of QGIS map canvas, showing the generated code and agent running information. Users can type their data requests, set the output directory, and press the 'Send Request' button to retrieve the data. The downloaded data will be automatically loaded into QGIS for further analysis. Note that the agent can also perform basic data operations and spatial analyses by leveraging the spatial programming ability of the backed GPT-4o model. Per our tests, it can conduct simple tasks such as spatial buffering and reprojection after data downloading, if these operations are specified in the user's request.

4. Case studies

The prototype agent currently supports six data sources, covering data types of vector, raster, and attribute. This section presents the results obtained by the agent when requesting data from each of these sources. We extensively tested the agent across various scenarios, resulting in a code repository with over 70 test cases. We encourage readers to explore these cases or to test their own data requests using either the QGIS plugin or the Python program.

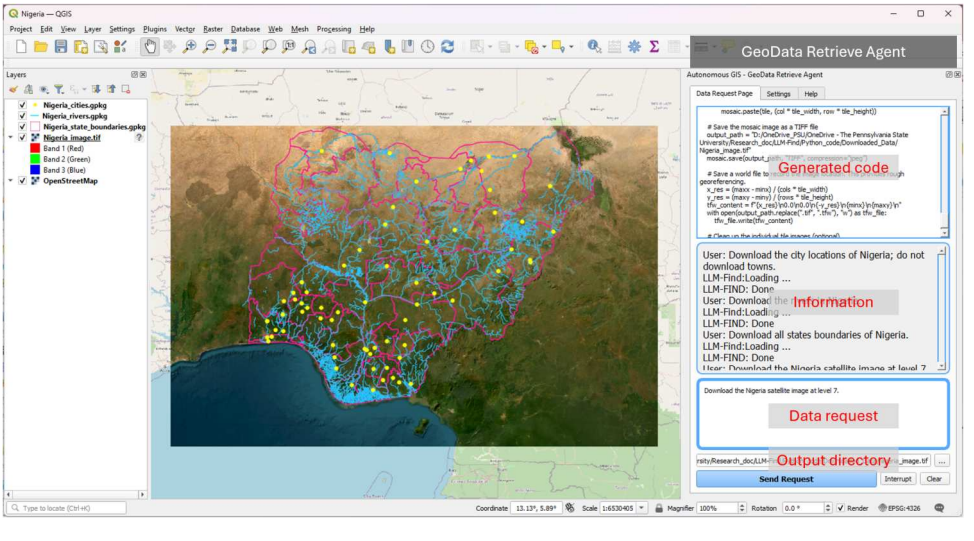


Figure 3. The QGIS plugin of the GeoData Retrieve Agent and the downloaded geospatial data of Nigeria, including cities (point), rivers (polyline), and state boundaries (polygon) from OpenStreetMap. The image basemap was downloaded from ESRI World Imagery using the plugin. Note that there are four individual data requests to retrieve the shown data (e.g. Download the rivers in Nigeria.).

4.1. Openstreetmap

OpenStreetMap is a collaborative project that creates a free, editable map of the world, built by volunteers who collect and contribute geographic data (Haklay and Weber 2008). It offers a vast repository of spatial data, including roads, trails, points of interest, and administrative boundaries, making it a valuable resource for various GIS applications. The data is structured using nodes, ways, and relations, which can be queried through the Overpass API. Our experiments showed that downloading data from OpenStreetMap was challenging, including generating correct Overpass queries and parsing results. The complexity of the OpenStreetMap data structure and tags increases the difficulties of fetching and saving the data for autonomous agents. For example, GPT-4o struggled to convert OpenStreetMap's *'relation'* data structure into multi-polygons with multi-holes. However, we significantly improved the success rate by providing a template program in the handbook. Figure 3 shows the results of downloading city points, river polylines, and state administrative polygons of Nigeria using the QGIS plugin. Figure 4 shows the four local data layers downloaded for the Pennsylvania State University in the US, including campus boundary, roads, buildings, and satellite imagery. Attributes were also downloaded as '*tags*' in OpenStreetMap as shown at the bottom of the map window.

4.2. U.S. Census data

U.S. Census data includes comprehensive demographic information about the population of the U.S., including variables such as population size, age distribution, income levels, education, and housing characteristics at varying geographic levels (e.g. blockgroup, census tract, county, and state). The agent handbook for the Census data includes the data access API information along with a template program to support the agent in selecting the needed variables from more than 27,000 available variables. Additionally, the U.S. Census Bureau's TIGER data (Topologically Integrated Geographic Encoding and Referencing) provides detailed geographic information about various administrative boundaries in the U.S., such as states, counties, and census tracts. These datasets

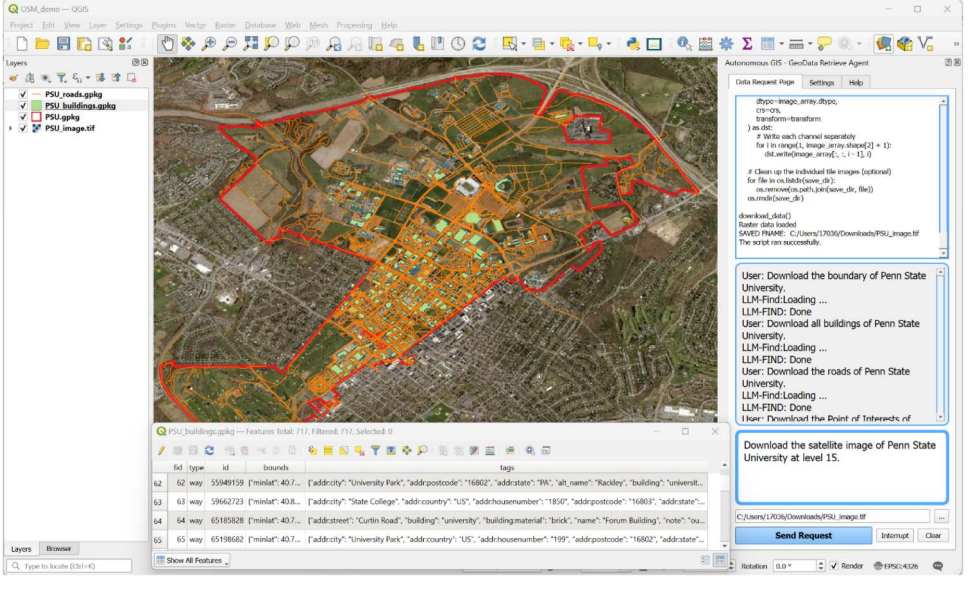


Figure 4. Four data layers downloaded for the Pennsylvania State University in the US, including campus boundary (request: Download the boundary of Penn State University.), buildings, roads, and satellite imagery (request: Download the satellite image of Penn State University at level 15). The attribute table for the building layer is also shown.

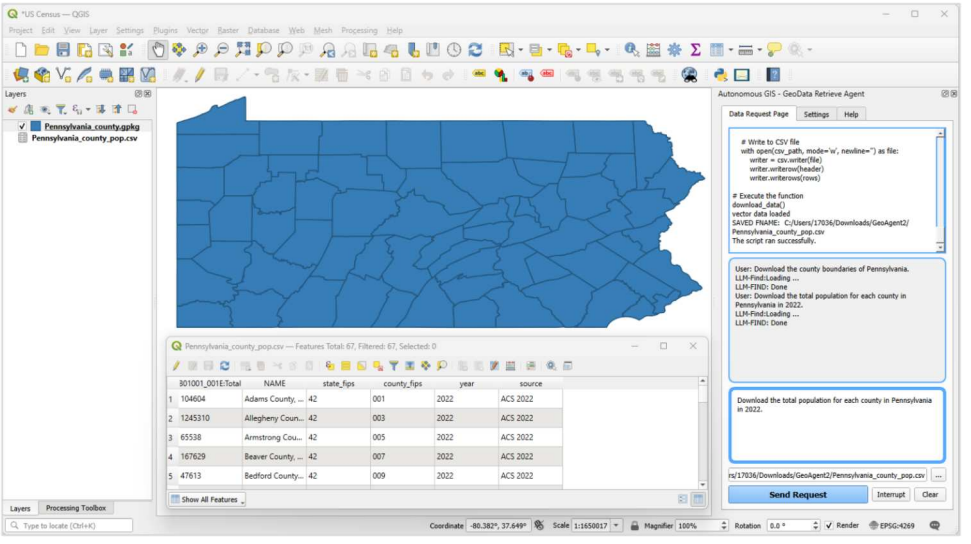


Figure 5. County boundaries of Pennsylvania downloaded with the QGIS plugin (request: Download the county boundaries of Pennsylvania.) and the population for each county in Pennsylvania in 2022. (request: Download the total population for each county in Pennsylvania in 2022).

are essential for geographic analysis and mapping. The handbook includes the boundary file URL structure for the agent to select the region and level, such as tract, county, or state, based on the request.

Figure 5 presents the county boundaries of Pennsylvania downloaded with the QGIS plugin and the population for each county in Pennsylvania in 2022. Figure 6 shows the results of another two requests: 'Download the population over 25 years old and the population with a college degree or higher at the state level of USA for 2012 and 2022' and 'Download the state boundary of the contiguous United States'. Multiple variables across multiple years can be retrieved in one request, as

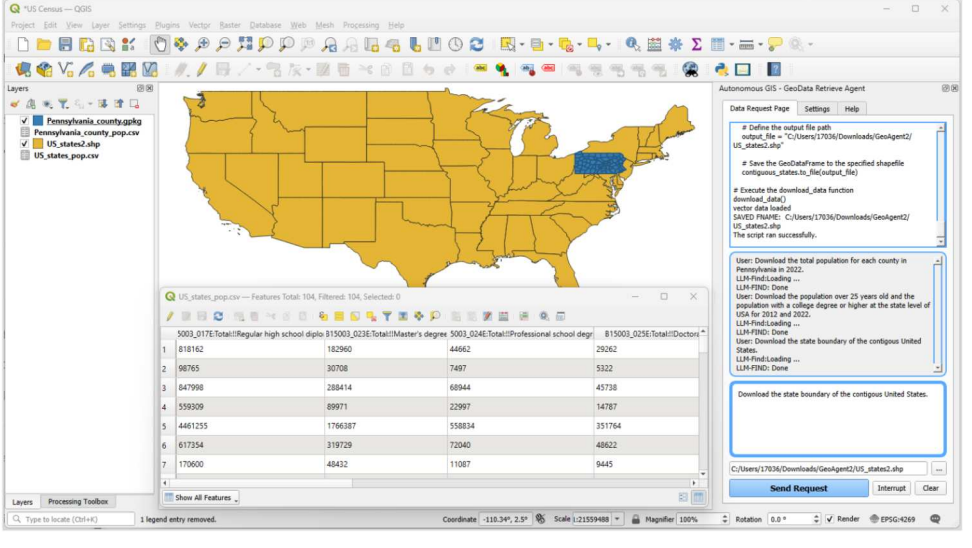


Figure 6. The downloaded state boundary of the contiguous United States and the population over 25 years old and the population with a college degree or higher at the state level of USA for 2012 and 2022.

shown in Figure 6. However, we observed that the agent may query the wrong variable combinations at times when the data involves multiple variables. For instance, when asking for the total senior population, it may only query the male or female population.

4.3. Satellite imagery

Satellite imagery is an essential type of Earth observation data used in both natural and social science research. However, retrieving satellite images is not trivial as researchers typically need a background in satellite sensors, accounts with data providers, and expertise in processing large volumes of image data. In this study, we equipped the prototype agent with relatively simple access to an image tile web service, i.e. ESRI World Imagery, enabling it to download images based on place names and bounding boxes without using API keys or accounts. The agent adopts the geocoding service provided by *nominatim.org* to retrieve the boundary from the given place name as the image extent. Future studies will explore integrating additional satellite imagery sources, such as NASA's Global Imagery Browse Services (GIBS) API, to provide access to global, full-resolution imagery.

Figure 7 shows the downloaded satellite images of the FAST Telescope in China (the top map), Yellowstone National Park (Map 1), a region defined by a bounding box (Map 2), Qingdao City in China (Map 3). Another two examples are shown in Figures 3 and 4, featuring the downloaded image of Nigeria and Pennsylvania State University. These examples, covering various geographic extents and resolution levels, demonstrate the feasibility of downloading satellite images with autonomous agents. New data sources along with corresponding handbooks can be added to the agent to enhance its capability in fetching other remote sensing images.

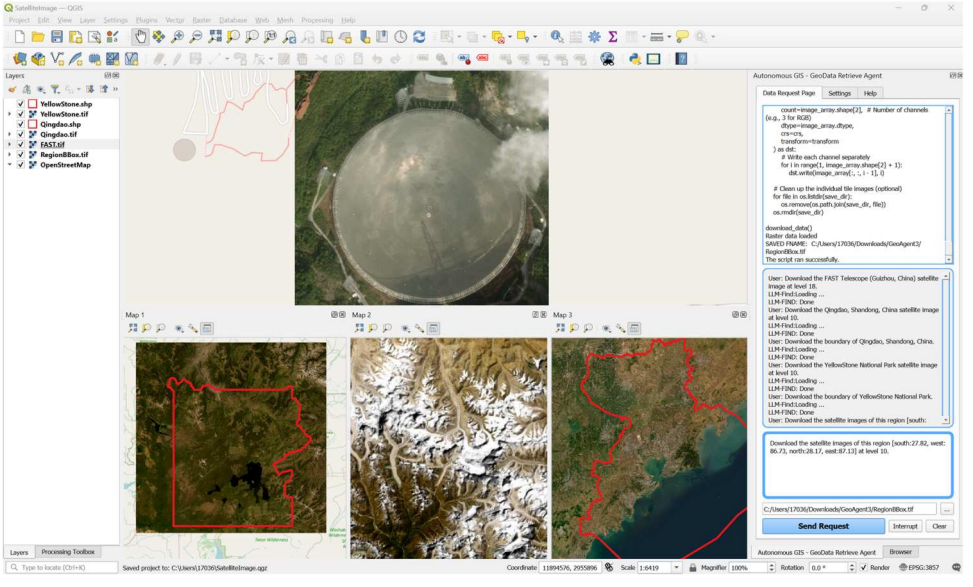


Figure 7. The downloaded satellite images. Top map: the FAST Telescope in China (request: Download the FAST Telescope (Guizhou, China) satellite image at level 18.). Map 1: Yellowstone National Park (request: Download the Yellowstone National Park satellite image at level 10.); Map 2: a region defined by a bounding box (request: Download the satellite images of this region [south:27.82, west:86.73, north:28.17, east:87.13] at level 10.); Map 3: Qingdao City in China (request: Download the Qingdao, Shandong, China satellite image at level 10). Note that the red boundaries in Map 1 and Map 3 were downloaded using the plugin by two separate requests based on the place names: Download the boundary of Yellowstone National Park and Download the boundary of Qingdao, Shandong, China.

4.4. Digital elevation model (DEM)

DEM is a 3D representation of a terrain surface in raster format, and it is an important environmental factor in various applications, such as ecology research and infrastructure planning. OpenTopography.org provides massive DEM products worldwide. The agent incorporates OpenTopography's global DEM products of 30m and 90m resolution and is able to retrieve DEM data based on a place name or bounding box. The top map of Figure 8 shows the downloaded DEM of Puerto Rico overlapped with three other data layers retrieved from OpenStreetMap including county boundaries, major rivers, and hospitals. In addition, the county level population and median household income of Puerto Rico in 2020 were also downloaded from the US Census using the plugin. The bottom map of Figure 8 shows the 30-meter resolution DEM of Chongqing, China downloaded with another request.

4.5. Weather data

We integrated the weather data API from OpenWeather (openweathermap.org) into the prototype agent. This API accepts location and period parameters to return historical, current, and forecast weather data. Note that the GPT-4o cannot get the current time on its own, so we added explicit prompts in the handbook to specify the period (historical, current, or forecast) or the current date and time to select appropriate API endpoints and parameters. Figure 9 shows an example of downloading historical weather data for Yulin City in Guangxi, China, during May 2024 using the Jupyter Notebook-based agent. Examples of fetching current weather data can be found in Figure 10. Since weather data is inherently spatiotemporal, this case study demonstrates that the agent can effectively manage both spatial and temporal aspects simultaneously, which is crucial in spatial analysis.

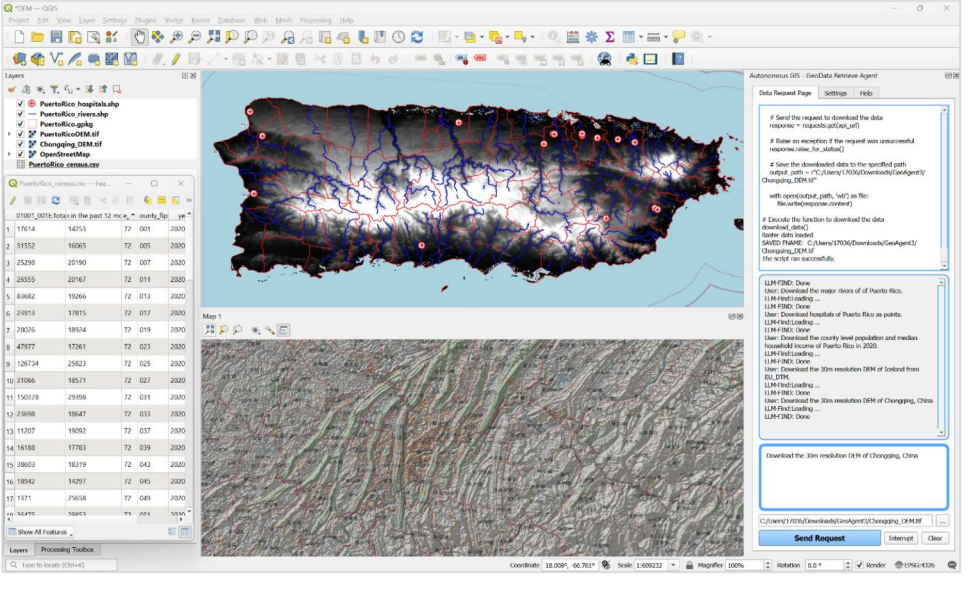
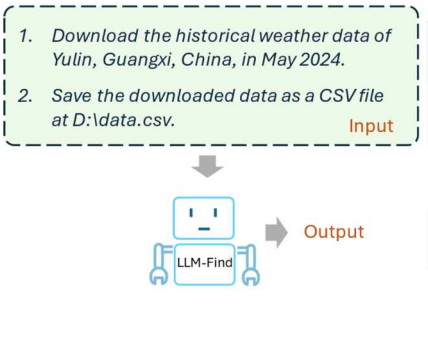


Figure 8. Top map: DEM of Puerto Rico overlapped with three other data layers retrieved from OpenStreetMap, including county boundaries, major rivers, and hospitals. (DEM request: Download the 90-meter resolution DEM data of Puerto Rico.) Bottom map: DEM of Chongqing, China (Request: Download the 30m resolution DEM of Chongqing, China). The DEM is displayed as Hillshade on top of the OpenStreetMap basemap with a transparency of 50%. Left table: the county level population and median household income of Puerto Rico in 2020 downloaded from the U.S. Census using the plugin.



place_name	date	hour	main_temp	main_feels_like	main_pressure	main_humidity
Yulin, Guangxi, China	2024-05-01	4	25.16	25.87	1008	82
Yulin, Guangxi, China	2024-05-01	5	25.41	26.15	1007	82
Yulin, Guangxi, China	2024-05-01	6	23.41	24.05	1008	86
Yulin, Guangxi, China	2024-05-01	7	23.09	23.70	1007	86
Yulin, Guangxi, China	2024-05-01	8	22.91	23.48	1007	85

Figure 9. Download the historical weather data for Yilin, Guangxi, China.

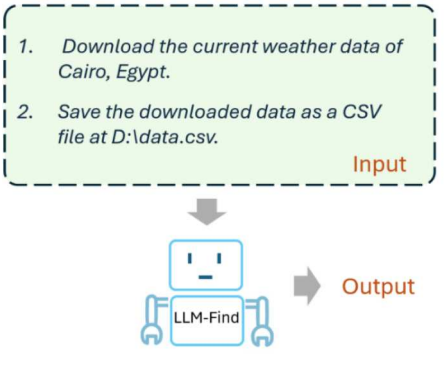
4.6. COVID-19 data

The New York Times provides a comprehensive dataset on COVID-19 accumulative cases, tracking the spread and impact of the virus across various regions (New York Times 2023). This dataset includes daily updates on case numbers, deaths, and other relevant metrics during the pandemic, making it a valuable resource for public health analysis and research. Similar to the U.S. Census Bureau administrative boundaries, this dataset can be accessed via URLs. We provided the agent with a handbook containing data URLs, data structures, and date ranges. As illustrated in Figure 11, the agent can download COVID-19 accumulative case data based on the requested geographic level and time period. This study case demonstrates the agent's ability to handle data sources without formal APIs.

5. Discussion

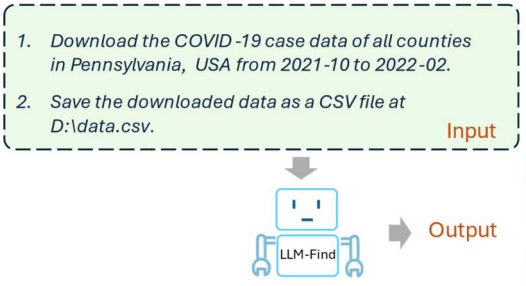
The proposed autonomous GIS agent framework, LLM-Find, demonstrated the feasibility of retrieving geospatial data using simple natural language input without human intervention. This research reflects the advocation of autonomous GIS (Li and Ning 2023), which envisions that the next generation of GIS could conduct spatial analysis with less or without human intervention.

The framework supports a plug-and-play system and handbook hubs as demonstrated by the prototype agent and case studies. First, this design is highly extendable: adding new data sources only requires updating the index and creating the corresponding handbook, with no impact on existing sources, which simplifies maintenance. The geospatial community can establish handbook



place_name	Cairo	wind_speed	4.12
date	2024-06-30	wind_deg	330
hour	16	clouds_all	0
main_temp	29.33	sys_type	1
main_feels_like	32.49	sys_id	2514
main_temp_min	29.03	sys_country	EG
main_temp_max	29.33	sys_sunrise	1719716222
main_pressure	1007	sys_sunset	1719766813
main_humidity	65	weather_id	800
main_sea_level	1007	weather_main	Clear
main_grnd_level	1001	weather_description	clear sky

Figure 10. Download current weather data for Cairo, Egypt.



date	county	state	fips	cases	deaths
2021-10-01	Adams	Pennsylvania	42001	12056	199.0
2021-10-01	Allegheny	Pennsylvania	42003	121400	2165.0
2021-10-01	Armstrong	Pennsylvania	42005	7676	170.0
2021-10-01	Beaver	Pennsylvania	42007	19753	432.0
2021-10-01	Bedford	Pennsylvania	42009	6031	154.0
...	...	...	...	...	...
2022-02-28	Washington	Pennsylvania	42125	50309	632.0
2022-02-28	Wayne	Pennsylvania	42127	9979	163.0
2022-02-28	Westmoreland	Pennsylvania	42129	79016	1335.0
2022-02-28	Wyoming	Pennsylvania	42131	5025	100.0
2022-02-28	York	Pennsylvania	42133	117826	1448.0

Figure 11. Download accumulative COVID-19 cases for counties in the US from October 2021 to February 2022.

hubs to share well-written and validated handbooks. Data providers should consider releasing data not only for human users but also for autonomous agents. Second, the proposed framework provides the flexibility to allow other agents to crawl online geospatial data sources, update the index and create handbooks autonomously. For example, LLM-Find can be connected to an *autonomous data discovery agent* powered by LLMs for Internet search, webpage analysis, geospatial data assessment, and handbook generation. The design principles of the proposed framework are universal and can be adopted by other GIS systems, such as ArcGIS (arcgis.com).

Given the various research subfields, topics, and applications in GIS, the involved data sources are highly diverse. In addition, the quality, quantity, and reliability of data sources often vary depending on the research questions. For example, while OpenStreetMap may have less road network coverage in remote areas in developing countries, it could still be a good data source for studies using city centroids and major river polylines. Thus, supporting customized handbooks is important to address such diverse needs. LLM-Find allows researchers to manually edit and add the handbook for a specific data source to access the data without further manual guidance. Researchers and data provider can also share their verified handbooks. The reliability, quality, and applicability of data sources can be documented in the handbook to facilitate the correct use for agents. We believe such experience sharing based on runnable programs can improve researchers' productivity by reducing the workload associated with data retrieval.

Despite the promising results, the current implementation of the framework has several limitations. One major limitation is that it only supports textual handbooks, which are unsuitable for some applications. For example, some geospatial data sources, particularly remote sensing images, have particular boundaries represented by vector files or raster maps. The framework needs further development to support multi-modal handbooks. Another limitation is the inability to support long handbooks due to the prompt length restrictions of LLMs. Some data sources require extensive materials to fetch data correctly. For example, the U.S. Census data provides more than 27,000 fine-grained variables for the American Community Survey 5-year survey (U.S. Census Bureau 2023), and NASA Global Imagery Browse Services APIs contain more than 1200 image layers. Retrieving the correct variable combinations or image layers from such a long and detailed list is challenging. A possible solution is to feed all descriptions of variables or layers and their into the handbook, leveraging LLMs that support large input tokens, such as Gemini 1.5 (Google 2024). However, computational time and cost may pose obstacles. Therefore, approaches such as Retrieval Augmented Generation (RAG) (Lewis et al. 2020) should be explored to support long and comprehensive handbooks.

Additionally, a data understanding and assessment module is necessary for the geospatial data retrieval agent. Human GIS analysts often choose between similar datasets or make compromises when encountering unsuitable datasets. For instance, they need to decide to use 1:5,000,000 or 1:50,000 scale boundaries for country-level analysis or select low-resolution terrain data when

high-resolution data is unavailable. Such decisions require experience, detailed knowledge of existing data, a profound understanding of application contexts, and creative problem-solving skills. Furthermore, the data understanding and assessment models is also needed to ensure the correctness of the retrieved data; it needs to 'see' the geospatial data's shape, boundary, attribute, location, and image quality to assess whether the results are correct. Future research could investigate whether autonomous agents can match or surpass human performance in executing these tasks.

As an LLM-based application, LLM-Find may have biases and errors. Since it is designed to retrieve data via programming, we consider the 'bias' in LLM-Find as the tendency to keep generating faulty code with similar errors due to limited domain knowledge in LLM. For example, GPT-4o was confused with the 'area' and 'relation' concepts in OpenStreetMap Overpass API. Our mitigating strategies include prompt engineering, providing template programs, and code review. The 'error' refers to retrieved data that does not satisfy a request due to the generated faulty code. We categorized the coding errors into two types: 1. unrunable fault code, triggering a Python runtime error, and 2. runnable faulty code, retuning incorrect or irrelevant data. For Type 1, the debugging module will send the error information, code, and data request to LLM for correction. This method usually works well in our tests. For Type 2, LLM-find currently lacks a data understanding module to verify whether the retrieved data matches the request or to address issues of data ambiguity or vagueness. For instance, OpenStreetMap data can be accessed using various methods via the Overpass Query Language with slightly different results, and the data lacks a universal attribute scheme or global standard and is subject to ongoing changes. To ensure the retrieved data is suitable for specific applications, we strongly recommend that users manually inspect it before proceeding with further analysis.

6. Conclusion

The proposed autonomous GIS agent framework has demonstrated the feasibility of retrieving geospatial data through simple natural language commands without human intervention. This capability indicates that such data retrieval agents can effectively provide necessary data to other agents for further analysis, supporting the vision of autonomous GIS. The framework's plug-and-play design and handbook hubs make it extendable and easy to maintain. Adding new data sources requires only updating the index and creating corresponding handbooks without affecting existing sources. This feature allows the geospatial community to share well-written and validated handbooks seamlessly. Moreover, the framework allows another agent to autonomously crawl online geospatial data sources, update the index, and create handbooks. This integration can enhance the autonomous data discovery process.

Based on the framework, we developed a proof-of-concept data retrieval agent with two implementations: a QGIS plugin named *GeoData Retrieve Agent* and a Python program in the Jupyter Notebook environment. The case studies demonstrated that the prototype agent could select the appropriate data source, load the associated handbook, and generate and execute a Python program to retrieve the requested geospatial data. Our study is among the first to develop an autonomous geospatial data retrieval agent. Future research should focus on enhancing autonomous online data discovery, assessment, and handbook generation to further advance autonomous GIS research.

Data and source code availability statement

The source code of the LLM-Find agent (Python program with Jupyter Notebook) along with over 70 data request examples can be found at https://github.com/gladcolor/LLM-Find. The source code of the QGIS plugin can be found at https://github.com/Teakinboyewa/AutonomousGIS_GeodataRetrieverAgent. The QGIS plugin can be downloaded from the official QGIS plugin webpage at https://plugins.qgis.org/plugins/AutonomousGIS_GeodataRetrieverAgent/#plugin-versions and installed following the User Manual.

Disclosure statement

No potential conflict of interest was reported by the author(s).

ORCID

Zhenlong Li http://orcid.org/0000-0002-8938-5466

References

Ageospatial. 2024. "GeoForge: Geospatial Analysis with Large Language Models (GeoLLMs)." *Medium*, February 20. https://medium.com/@ageospatial/geoforge-geospatial-analysis-with-large-language-models-geollms-2d3a0eaff8aa

Akinboyewa, T., Z. Li, H. Ning, and M. N. Lessani. 2024. "GIS Copilot: Towards an Autonomous GIS Agent for Spatial Analysis". arXiv:2411.03205. https://doi.org/10.48550/arXiv.2411.03205

Akinboyewa, T., H. Ning, M. N. Lessani, and Z. Li. 2024. "Automated Floodwater Depth Estimation Using Large Multimodal Model for Rapid Flood Mapping." *Computational Urban Science* 4 (1): 12. https://doi.org/10.1007/s43762-024-00123-3.

Anthropic. 2024. "Introducing Computer Use, a New Claude 3.5 Sonnet, and Claude 3.5 Haiku." October 22. https://www.anthropic.com/news/3-5-models-and-computer-use

Bryson, T. 2024. "From Questions to Discoveries: NASA's New Earth Copilot Brings Microsoft AI Capabilities to Democratize Access to Complex Data." The Official Microsoft Blog, November 14. https://blogs.microsoft.com/blog/2024/11/14/from-questions-to-discoveries-nasas-new-earth-copilot-brings-microsoft-ai-capabilities-to-democratize-access-to-complex-data/

Crooks, A., and Q. Chen. 2024. "Exploring the new Frontier of Information Extraction through Large Language Models in Urban Analytics." *Environment and Planning B: Urban Analytics and City Science* 51 (3): 565–569. https://doi.org/10.1177/23998083241235495.

Feng, Y., L. Ding, and G. Xiao. 2023. "GeoQAMap-Geographic Question Answering with Maps Leveraging LLM and Open Knowledge Base (Short Paper)." *12th International Conference on Geographic Information Science (GIScience 2023)*. https://drops.dagstuhl.de/opus/volltexte/2023/18923/

Ghaffary, S., and R. Metz. 2024. "OpenAI Nears Launch of AI Agent Tool to Automate Tasks for Users." Bloomberg.Com, November 13. https://www.bloomberg.com/news/articles/2024-11-13/openai-nears-launch-of-ai-agents-to-automate-tasks-for-users

Gispo. 2021. *How Big is the QGIS Community?* Gispo. https://www.gispo.fi/en/blog/how-big-is-the-qgis-community/

Google. 2024. "Our Next-Generation Model: Gemini 1.5." Google. February 15, 2024. https://blog.google/technology/ai/google-gemini-next-generation-model-february-2024/.

Grant, T. B., J. Goldwater, and E. Knudsen. 2024 "Can Language Models Solve Complex Subsurface Data Integrations: Building Subsurface Copilots with Large Language Models (LLMs)," Fourth EAGE Digitalization Conference and Exhibition, Paris, France. March 25–27, 2024:1–5. European Association of Geoscientists & Engineers. https://doi.org/10.3997/2214-4609.202439022

Gupta, D. V., A. S. A. Ishaqui, and D. K. Kadiyala. 2024. "Geode: A Zero-shot Geospatial Question-Answering Agent with Explicit Reasoning and Precise Spatio-Temporal Retrieval." arXiv:2407.11014. https://doi.org/10.48550/arXiv.2407.11014

Haklay, M., and P. Weber. 2008. "Openstreetmap: User-Generated Street Maps." *IEEE Pervasive Computing* 7 (4): 12–18. https://doi.org/10.1109/MPRV.2008.80

Hao, Y., J. Qi, X. Ma, S. Wu, R. Liu, and X. Zhang. 2024. "An LLM-Based Inventory Construction Framework of Urban Ground Collapse Events with Spatiotemporal Locations." *ISPRS International Journal of Geo-Information* 13 (4): 133. https://doi.org/10.3390/ijgi13040133.

Hong, S., L. Xiao, X. Zhang, and J. Chen. 2024. "ArgMed-Agents: Explainable Clinical Decision Reasoning with Large Language Models via Argumentation Schemes" arXiv:2403.06294. https://doi.org/10.48550/arXiv.2403.06294

Hu, Y., G. Mai, C. Cundy, K. Choi, N. Lao, W. Liu, G. Lakhanpal, R. Z. Zhou, and K. Joseph. 2023. "Geo-Knowledge-Guided GPT Models Improve the Extraction of Location Descriptions from Disaster-Related Social Media Messages." *International Journal of Geographical Information Science* 37 (11): 2289–2318. https://doi.org/10.1080/13658816.2023.2266495.

Kim, H., and S. Lee. 2024. "POI GPT: Extracting POI Information from Social Media Text Data." *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences, XLVIII-4-W10-2024*, 113–118. ISPRS WG IV/98th International Conference on Smart Data and Smart Cities (SDSC) - 4–7 June 2024, Athens, Greece. https://doi.org/10.5194/isprs-archives-XLVIII-4-W10-2024-113-2024

Lewis, P., E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, et al. 2020. "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." *Advances in Neural Information Processing Systems* 33:9459–9474.

Li, Z., and H. Ning. 2023. "Autonomous GIS: The Next-Generation AI-Powered GIS." *International Journal of Digital Earth* 16 (2): 4668–4686. https://doi.org/10.1080/17538947.2023.2278895.

Li, D., and Z. Zhang. 2023. "MetaQA: Enhancing Human-Centered Data Search Using Generative Pre-Trained Transformer (GPT) Language Model and Artificial Intelligence." *PLoS One* 18 (11): e0293034. https://doi.org/10.1371/journal.pone.0293034.

Ma, K., S. Zheng, M. Tian, Q. Qiu, Y. Tan, X. Hu, H. Li, and Z. Xie. 2023. "CnGeoPLM: Contextual Knowledge Selection and Embedding with Pretrained Language Representation Model for the Geoscience Domain." *Earth Science Informatics* 16 (4): 3629–3646. https://doi.org/10.1007/s12145-023-01112-6.

Mansourian, A., and R. Oucheikh. 2024. "ChatGeoAI: Enabling Geospatial Analysis for Public through Natural Language, with Large Language Models." *ISPRS International Journal of Geo-Information* 13 (10): 348. https://doi.org/10.3390/ijgi13100348.

Menezes, R. 2023. "Beyond Keywords: Comparing Insights from Unstructured Data using Generative GPT Search and Hybrid Extractive Search in Petroleum Exploration." EAGE Workshop on Data Science - From Fundamentals to Opportunities, Kuala Lumpur, Malaysia, October 17–18, 2023. European Association of Geoscientists & Engineers, p.1–8. https://doi.org/10.3997/2214-4609.202377039

Mohanad Diab. 2024. "Beyond Maps: Advancing Geospatial AI with Multimodal Foundation Models." https://www.politesi.polimi.it/handle/10589/218643

Mosser, L., P. Aursand, K. S. Brakstad, C. Lehre, and J. Myhre-Bakkevig. 2024. "Exploration Robot Chat: Uncovering Decades of Exploration Knowledge and Data with Conversational Large Language Models." April 17. https://doi.org/10.2118/218439-MS

New York Times. 2023. "Coronavirus (Covid-19) Data in the United States (Archived)." The New York Times." *New York Times*. https://github.com/nytimes/covid-19-data.

OpenAI. 2024. OpenAI Platform, June. https://platform.openai.com

OpenStreetMap. 2024. Overpass API, August 20. https://wiki.openstreetmap.org/wiki/Overpass_API

Raffel, C., N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. 2020. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer." *Journal of Machine Learning Research* 21 (140): 1–67.

Staniek, M., R. Schumann, M. Züfle, and S. Riezler. 2023. "Text-to-OverpassQL: A Natural Language Interface for Complex Geodata Querying of OpenStreetMap." arXiv:2308.16060. https://doi.org/10.48550/arXiv.2308.16060

Sun, X., H. Meng, S. Chakraborty, A. S. Bedi, and A. Bera. 2024. "Beyond Text: Utilizing Vocal Cues to Improve Decision Making in LLMs for Robot Navigation Tasks." arXiv:2402.03494. https://doi.org/10.48550/arXiv.2402.03494

Tao, R., and J. Xu. 2023. "Mapping with ChatGPT." *ISPRS International Journal of Geo-Information* 12 (7): 284. https://doi.org/10.3390/ijgi12070284.

Touvron, H., L. Martin, K. Stone, P. Albert, A. Almahairi, Y. Babaei, N. Bashlykov, et al. 2023. "Llama 2: Open Foundation and Fine-Tuned Chat Models" arXiv:2307.09288. https://doi.org/10.48550/arXiv.2307.09288

US Census Bureau.2023. "Census Data API User Guide." Census.Gov. December 21, 2023. https://www.census.gov/data/developers/guidance/api-user-guide.html.

US Census Bureau. 2024. "TIGER/Line Shapefiles." Census.Gov, October 9. https://www.census.gov/geographies/mapping-files/time-series/geo/tiger-line-file.html

Wang, S., T. Hu, H. Xiao, Y. Li, C. Zhang, H. Ning, R. Zhu, Z. Li, and X. Ye. 2024. "GPT, Large Language Models (LLMs) and Generative Artificial Intelligence (GAI) Models in Geospatial Science: A Systematic Review." *International Journal of Digital Earth* 17 (1). https://doi.org/10.1080/17538947.2024.2353122.

Zhang, Y., Z. He, J. Li, J. Lin, Q. Guan, and W. Yu. 2024b. "MapGPT: An Autonomous Framework for Mapping by Integrating Large Language Model and Cartographic Tools." https://doi.org/10.13140/RG.2.2.24858.62407

Zhang, Y., C. Wei, Z. He, and W. Yu. 2024a. "GeoGPT: An Assistant for Understanding and Processing Geospatial Tasks." *International Journal of Applied Earth Observation and Geoinformation* 131:103976. https://doi.org/10.1016/j.jag.2024.103976.

Zhao, P., H. Zhang, Q. Yu, Z. Wang, Y. Geng, F. Fu, L. Yang, W. Zhang, J. Jiang, and B. Cui. 2024. "Retrieval-Augmented Generation for AI-Generated Content: A Survey." arXiv.Org,, February 29. https://arxiv.org/abs/2402.19473v6

Appendix 1. An example prompt for selecting the data source

Your role: A professional Python programmer in geographic information science (GIScience). You have worked on GIScience for more than 20 years and know every detail and pitfall when collecting data and coding. You know which websites you can get suitable spatial data and know the methods or tricks to download data, such as OpenStreetMap, Census Bureau, or various APIs. You are also experienced in processing the downloaded data, including saving them in suitable formats, map projections, and creating detailed and useful meta-data.

Your mission: select a suitable data source from the given list to download the requested geospatial data for this task:

1. Download all province boundaries of China mainland.
2. Save the downloaded data as polygons in GeoPackage format at: E:\China_mainland_Province_boundary.gpkg

Requirements:
1. Return the exact name of the data source as the given names.
2. If a data source is given in the task, e.g. OpenStreetMap or Census Bureau, you need to select that given data source.
3. If you need to download the administrative boundary of a place without mentioning the data sources, you can get data from OpenStreetMap.If you need to download the US Census tract and block group boundaries, download them from Census Bureau. Follow the given JSON format.
4. If you cannot find a suitable data source in the given sources, return a data source you think is most appropriate.
5. DO NOT make fake data source. If you cannot find any suitable data source, return 'Unknown' as for the 'Selected data source' key in the reply JSON format. DO NOT use '"json and"'

Data sources:
1. OpenStreetMap. You can download the administrative boundaries, street networks, points of interest (POIs) from OpenStreetMap.
2. US Census Bureau boundary. It provides the US administrative boundaries (nation, state, county, tract, and block group level) and metropolitan statistic areas.
3. US Census Bureau demography. It provides the demographic and socio-economic data, such as population, gender, income, education, and race.
4. US COVID-19 data by New York Times. Cumulative counts of COVID-19 cases and deaths in the United States, at the state and county level, over time from 2020-01-21 to 2023-03-23.
5. OpenWeather data. It provides historical, current, and forecast weather data. The historical data can be back to 2023-08. API limited: [Hourly forecast: 4 days, Daily forecast: 16 days, 3 h forecast: 5 days]
6. ESRI World Imagery (for export). It is a web map service, providing satellite image tiles. You can download tiles and mosaic them into a large image.

Your reply example: {'Explanation': "According to the use requests of US state administrative boundary from OpenStreetMap, I should download data from OpenStreetMap.", "Selected data source": 'OpenStreetMap'}

Appendix 2. An example prompt for fetching administrative boundary data

Your role: A professional Python programmer in geographic information science (GIScience). You have worked on GIScience for more than 20 years and know every detail and pitfall when collecting data and coding. You know which websites you can get suitable spatial data and know the methods or tricks to download data, such as OpenStreetMap, Census Bureau, or various APIs. You are also experienced in processing the downloaded data, including saving them in suitable formats, map projections, and creating detailed and useful meta-data. When downloading geospatial data, the technical handbook for a particular data source is provided; you can follow it, and write Python code carefully to download the data.

Your mission: download geospatial data from the given data source for this task:

1. Download all province boundaries of China mainland.
2. Save the downloaded data as polygons in GeoPackage format at: E:\dwonloaded_data.gpkg

Technical handbook:

1. If the requested area is given in an English name, you need to use `['name:en'='XX']` to filter the place in Overpass queries; otherwise you will get empty results. The `name` tag in OpenStreetMap usually is in the location language.
2. If you need to download the administrative boundary of a place from OpenStreetMap, please use a Python package named 'OSMnx' by this code line: `ox.geocode_to_gdf(query, which_result=None, by_osmid=False, buffer_dist=None)`. This method is fast.
3. If you need to download POIs, you may use the Overpass API, which is faster than OSMnx library. Code example is: `area['ISO3166-2'='US-PA']->.searchArea;(nwr[amenity='hospital'](area.searchArea););out center;`
4. If you need to download polylines, you may use the Overpass API, which is faster than OSMnx library.
5. If you need to use a boundary to filter features in GeoPandas, this is the code: `gpd.sjoin(gdf, boundary, how='inner', op='within')`.
6. If you need to download multiple administrative boundaries at the same level, e.g., states or provinces, DO NOT use OSMnx because it is slow. You can use Overpass API. Example code: `area['ISO3166-1'='US'][admin_level=2]->.us;(relation(area.us)['admin_level'='4'];);out geom;`. Overpass API is quicker and simpler; you only need to carefully set up the administrative level.
7. Only use OSMnx to obtain the place boundaries; do no use it to download networks or POIs as it is very slow! Instead, use Overpass Query (endpoint: https://overpass-api.de/api/interpreter).
8. If using Overpass API, you need to output the geometry, i.e., using `out geom;` in the query. The geometry can be accessed by `returned_json['elements']['geometry']`; the gemotry is a list of points as `{'lat': 30.5, 'lon': 114.2}`.
9. Use GeoPandas, rather than OSGEO package, to create vectors.
10. If the file saving format is not given in the tasks, save the downloaded files into GeoPackage format.
11. You need to create Python code to download and save the data. Another program will execute your code directly.
12. Put your reply into a Python code block, explanation or conversation can be Python comments at the beginning of the code block(enclosed by ```python and```).
13. The download code is only in a function named 'download_data()'. The last line is to execute this function.
14. When downloading OSM data, no need to use 'building' tags if it is not asked for.
15. You need to keep most attributes of the downloaded data, such as place name, street name, road type, and level.
16. Throw an error if the the program fails to download the data; no need to handle the exceptions.
17. If you need to convert the OpenStreetMap returned JSON to GeoJSON, you can add this line to the OverPass query: `item ::=::,::geom=geom(),_osm_type=type(), ::id=id();`. Note the converted GeoJSON may only contains polygons, no polygons.

This is a program for your reference; note that you can improve it:

```
# Below is a program to download the province boundaries of Cuba.
import geopandas as gpd
import pandas as pd
import requests
import json
from shapely.ops import linemerge, unary_union, polygonize
from shapely.geometry import MultiLineString, Polygon, MultiPolygon, LineString
from shapely.ops import polygonize
def download_data():

    # Define Overpass API query to download province boundaries of Cuba
    overpass_url = "https://overpass-api.de/api/interpreter"
    overpass_query = """
    [out:json];
    area["ISO3166-1"="CU"][admin_level=2]->.cu;
    relation(area.cu)["admin_level"="4"];
    out geom;
    """
    # Send request to Overpass API
    response = requests.get(overpass_url, params={'data': overpass_query})
    response.raise_for_status() # Automatically raises an error for bad status codes
    data = response.json()
```

```
    # Parse the JSON response
    property_list = []
    geometry_list = []
    for element in data['elements']: # each province
        way_list = []
        outer_lines = []
        inner_lines = []
        for member in element.get('members', []): # each way/polyline
            if 'geometry' in member:
                if member['type'] == 'way':
                    way_points = [(point['lon'], point['lat']) for point in member['geometry']]
                    line_string = LineString(way_points)
                    if member['role'] == 'outer':
                        outer_lines.append(line_string)

                    if member['role'] == 'inner':
                        inner_lines.append(line_string)

        # Create polygon. We use Multi-polygon to represent all polygons
        merged = linemerge([*outer_lines]) # merge LineStrings
        borders = unary_union(merged) # linestrings to a MultiLineString
        outer_polygons = list(polygonize(borders))
        outer_polyon = MultiPolygon(outer_polygons)
        if len(inner_lines) > 0:
            merged = linemerge([*inner_lines]) # merge LineStrings
            borders = unary_union(merged) # linestrings to a MultiLineString
            inner_polyons = list(polygonize(borders))
            inner_polyon = MultiPolygon(inner_polyons)
            final_polygon = outer_polyon.difference(inner_polyon)
        else:
            final_polygon = outer_polyon
        geometry_list.append(final_polygon)
        # extract the properties
        properties = {
            key: ', '.join(map(str, value)) if isinstance(value, list) else str(value)
            for key, value in element.items() if key not in {'geometry', 'members'}
        }
        property_list.append(properties)
    df = pd.DataFrame.from_dict(property_list)
    gdf = gpd.GeoDataFrame(df, geometry=geometry_list)
    gdf.crs = 'EPSG:4326'
    # # Save to GeoPackage
    output_file = r"E:\Cuba_Province_boundary.gpkg"
    gdf.to_file(output_file, layer='province_boundaries', driver='GPKG')

download_data()
```