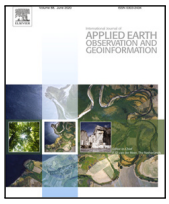
International Journal of Applied Earth Observation and Geoinformation 136 (2025) 104312

Contents lists available at ScienceDirect

International Journal of Applied Earth Observation and Geoinformation

journal homepage: www.elsevier.com/locate/jag



Geospatial large language model trained with a simulated environment for generating tool-use chains autonomously

Yifan Zhang [a], Jingxuan Li [a], Zhiyun Wang [a], Zhengting He [a], Qingfeng Guan [a], Jianfeng Lin [b], Wenhao Yu [a,*]

[a] School of Geography and Information Engineering, China University of Geosciences, Wuhan, China
[b] Meituan, Beijing, China

ARTICLE INFO

Dataset link: https://github.com/AGI-GIS/GTChain

Keywords:
GIS
Large language model
Tool-use chain

ABSTRACT

Solving geospatial tasks generally requires multiple geospatial tools and steps, i.e., tool-use chains. Automating the geospatial task solving process can effectively enhance the efficiency of GIS users. Traditionally, researchers tend to design rule-based systems to autonomously solve similar geospatial tasks, which is inflexible and difficult to adapt to different tasks. With the development of Large Language Models (LLMs), some research suggests that LLMs have the potential for intelligent task solving with their tool-use ability, which means LLMs can invoke externally provided tools for specific tasks. However, most studies rely on closed-source commercial LLMs like ChatGPT and GPT-4, whose limited API accessibility restricts their deployment on local private devices. Some researchers in the general domain proposed using instruction tuning to improve the tool-use ability of open-source LLMs. However, the requirement of tool-use chains to solve geospatial tasks, including multiple data input and output processes, poses challenges for collecting effective instruction tuning data. To solve these challenges, we propose a framework for training a Geospatial large language model to generate Tool-use Chains autonomously (GTChain). Specifically, we design a seed task-guided self-instruct strategy to generate a geospatial tool-use instruction tuning dataset within a simulated environment, encompassing diverse geospatial task production and corresponding tool-use chain generation. Subsequently, an open-source general-domain LLM, LLaMA-2-7B, is fine-tuned on the collected instruction data to understand geospatial tasks and learn how to generate geospatial tool-use chains. Finally, we also collect an evaluation dataset to serve as a benchmark for assessing the geospatial tool-use ability of LLMs. Experimental results on the evaluation dataset demonstrate that the fine-tuned GTChain can effectively solve geospatial tasks using the provided tools, achieving 32.5% and 27.5% higher accuracy in the percentage of correctly solved tasks compared to GPT-4 and Gemini 1.5 Pro, respectively.

Contents

* Corresponding author.
E-mail addresses: yifanzhang@cug.edu.cn (Y. Zhang), lijingxuan@cug.edu.cn (J. Li), wangzhiyun@cug.edu.cn (Z. Wang), zhengtinghe@cug.edu.cn (Z. He), guanqf@cug.edu.cn (Q. Guan), linjianfeng03@meituan.com (J. Lin), yuwh@cug.edu.cn (W. Yu).

https://doi.org/10.1016/j.jag.2024.104312
Received 12 September 2024; Received in revised form 26 November 2024; Accepted 4 December 2024
Available online 20 December 2024

1. Introduction

Geographic Information Systems (GIS) have proven to be highly effective in supporting research across various domains (Maguire, 1991), including urban planning (Yeh, 1999; Albert et al., 2014), transportation management (Goodchild, 2000), and disaster response (Franch-Pardo et al., 2020; Zerger and Smith, 2003; Jamhiri et al., 2023a,b). To meet diverse spatial analysis requirements, numerous algorithms and models have been developed and packaged as geospatial tools that are easily accessible to relevant users (Wright et al., 1997). Generally, users need to combine multiple geospatial tools and follow a series of steps to address a geospatial task. In other words, solving geospatial tasks often requires tool-use chains, where outputs from earlier tools or steps serve as prerequisites for subsequent ones. The necessity of tool-use chains means users must perform multiple operations to complete geospatial tasks. Therefore, automating the geospatial task solving process can significantly enhance the efficiency of GIS users (Gao and Goodchild, 2013; Zhu et al., 2021).

Traditionally, researchers tend to design rule-based systems to autonomously solve similar geospatial tasks (Abdella and Alfredsen, 2010). For example, ArcGIS provides the function of ModelBuilder, which enables users to design tool chains that connect multiple geospatial tools in sequence (Mericskay, 2018; Allen, 2011). In this way, users only need to input the necessary spatial datasets, and corresponding spatial analysis tasks can be solved by sequentially using the selected tools. However, such a strategy is inflexible and difficult to adapt to different tasks. Specifically, even a minor change in a geospatial task, such as requiring one more or one fewer tool, necessitates revisions to the designed chain. Therefore, exploring effective and flexible strategies to automate the geospatial task solving process is a focused problem in the GIS domain (Scheider et al., 2021; Gao, 2020; Yuan et al., 2019; Li and Ning, 2023).

Recently, with the development of Large Language Models (LLMs), some research suggests that LLMs have the potential for intelligent task solving (Guo et al., 2024; Wang et al., 2024c). Trained on extensive corpora, LLMs demonstrate robust semantic understanding and reasoning abilities, achieving excellent performance on various natural language processing tasks (Peng et al., 2024). Moreover, although LLMs may not excel in professional tasks (e.g., performing complex mathematical calculations), researchers have found that LLMs can solve such tasks by utilizing external tools, such as a calculator (Yao et al., 2022). In other words, while LLMs may not solve complex tasks directly, their robust understanding and reasoning capabilities enable them to effectively use external tools. This tool-use ability of LLMs has surprised researchers, and studies across different domains, including computer vision (Wu et al., 2023), disease (Dai et al., 2023), and GIS (Zhang et al., 2024e,c), suggest that LLMs can effectively invoke external tools for specific tasks.

Most of these studies rely on closed-source commercial LLMs, such as ChatGPT and GPT-4, which exhibit robust tool-use abilities. However, these models can only be accessed via APIs, which restricts their deployment on local private devices (Zhang et al., 2024d). To mitigate this gap, some researchers in the general domain proposed using instruction tuning to improve the tool-use ability of open-source LLMs (Qin et al., 2023; Jiang et al., 2024b). For example, Yang et al. (2024) proposed collecting an instruction tuning dataset relevant to image processing tools, which can be used to train open-source LLMs, such as LLaMA and OPT, to use these tools. Specifically, each case in the dataset includes a question (or task) and its corresponding answer (or solution). Instruction tuning aims to align the output of LLMs with the corresponding answer based on the question, which has been validated as an effective strategy to improve the performance of LLMs (Deng et al., 2024; Zhang et al., 2024a).

However, different from tasks in the general domain (e.g., image processing), solving geospatial tasks generally requires tool-use chains, including multiple data input and output processes, which poses challenges for collecting effective instruction tuning data. An illustrative example is presented in Fig. 1. Given a common geospatial task (e.g., school site selection), three different tools (Buffer, Intersect, and Erase) and five steps are required to solve it. Moreover, the output from earlier steps generally serves as the input for later steps, forming a tool-use chain. However, a typical image processing task (e.g., image segment) only needs one tool. Therefore, collecting instruction tuning data for geospatial tasks, including a wide variety of geospatial task scenarios and their corresponding effective tool-use chain solutions, is a critical problem.

To address these challenges, a seed task-guided self-instruct strategy is designed in this paper to generate a geospatial tool-use instruction tuning dataset within a simulated environment. Specifically, for a given geospatial tool, an example task is first manually crafted by GIS professionals. Subsequently, the example tasks of multiple geospatial tools serve as seed tasks to guide a robust LLM (GPT-4 is used in this paper) to generate relevant geospatial tasks. This approach enables the

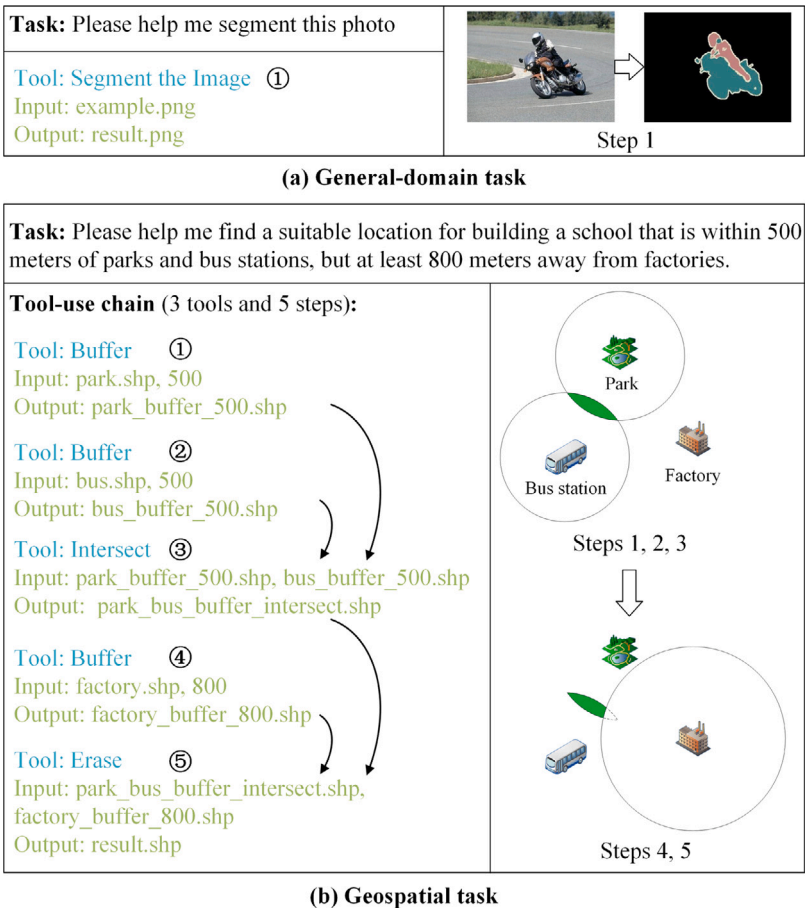


Fig. 1. The challenge of collecting instruction tuning data for geospatial tasks. Compared to general-domain tasks (a), it generally requires multiple tools and steps to solve geospatial tasks (b).

generation of a wide variety of geospatial task scenarios. However, it is labor-intensive to manually generate a corresponding tool-use chain for each geospatial task, as illustrated in Fig. 1. To alleviate this, we also design a simulated environment to mimic the process of tool execution and data stream. Specifically, given an input data path and a tool name, a robust LLM is guided to simulate its output data path (similar to the path in Fig. 1), rather than executing the tool to obtain the output. In this way, both geospatial tasks and their corresponding tool-use chains can be obtained.

Based on the collected instruction tuning data, we propose a framework for training a **G**eospatial LLM to autonomously generate **T**ool-use **Chain**s, called GTChain. Specifically, an open-source general-domain LLM, LLaMA-2-7B (Touvron et al., 2023), is fine-tuned using the collected instruction data to understand geospatial tasks and learn how to generate geospatial tool-use chains. Additionally, we collect an evaluation dataset to serve as a benchmark for assessing the geospatial tool-use capabilities of LLMs. Unlike the instruction tuning data, the evaluation dataset is labeled by GIS professionals. Experimental results on the evaluation dataset demonstrate that the fine-tuned LLM can effectively solve geospatial tasks using the provided tools, achieving 32.5% and 27.5% higher accuracy compared to GPT-4 and Gemini 1.5 Pro, respectively.

2. Related work

Since its inception, GIS has evolved rapidly. Nowadays, many GIS packages are available to support users with their spatial analysis needs (Chu et al., 2024; Zhang et al., 2022). As the volume of location data continues to increase, the demand for GIS in spatial analysis has become both widespread and essential (Goodchild, 2016; Yue and Jiang, 2014). Additionally, researchers often need to process spatial data iteratively, creating a need for automated or intelligent GIS systems that can alleviate workload and enhance efficiency. Researchers from various fields have paid attention to this challenge and proposed many solutions.

For example, during the scripting stage, researchers proposed to compile frequently used sequences of GIS commands tailored for specific applications or analytical tasks into scripts that function as macro-commands (Clerici et al., 2006; Abdella and Alfredsen, 2010). These scripts enable analysts to repeatedly reuse their analyses and share their work with other users. After that, some GIS software provides users the platform to design their workflows, such as the ModelBuilder in ArcGIS and the Graphic Modeler in QGIS (Magesh et al., 2012; Park et al., 2019). With these workflows, similar geospatial tasks can be easily solved by repeating them with only input changes (e.g., data and parameters). However, these workflows are fixed, and cannot be adapted to other geospatial tasks.

With the advancement of deep learning and the commercial deployment of LLMs such as ChatGPT, there is growing excitement about the potential for artificial intelligence, and even Artificial General Intelligence, to become a reality (Janowicz et al., 2020; Mai et al., 2024; Ge et al., 2024). This growing enthusiasm has spurred researchers to explore the use of large models across various professional domains (Jiang et al., 2024a). For example, visual language models and generative diffusion models have been successfully applied to zero-shot SAR target recognition, achieving promising results (Wang et al., 2024b). In the geoscience field, Zhao et al. (2024) provide an in-depth analysis of the progress, challenges, and future directions of AI,

"Tool_Name":"Buffer",
"Function":"Creates a buffer zone around a shapefile.",
"Input":"Shapefile path, buffer radius.",
"Output":"Path of the buffered shapefile.",
"Task Example":"Please help me find areas that are located within a maximum distance of 500 meters from subway stations. The file of subway stations is located at ./subway_station.shp."

Fig. 2. The manually crafted template for each tool (Buffer as an example).

emphasizing both its potential and its limitations. Chen et al. (2024) advocate for a balanced integration of AI advancements with domain-specific expertise, highlighting the critical need for cross-disciplinary collaboration. Moreover, since the robust semantic understanding, reasoning, and planning capabilities of LLMs, researchers have also be prompted to explore their applicability in solving geospatial tasks. For instance, Hu et al. (2023) proposed utilizing LLMs for extracting location descriptions from disaster-related social media messages, while Mai et al. (2024) investigated the effectiveness of LLMs on four classical geospatial tasks. Despite these promising developments, the application of LLMs remains predominantly limited to text-related tasks.

To address this, some studies have proposed extending the application scenarios of LLMs by leveraging their tool-use capabilities (Zhuang et al., 2024; Schick et al., 2024). Specifically, rather than directly outputting results, LLMs can be employed to understand tasks and plan the use of specific tools to address them (Wei et al., 2024). For example, if LLMs are not proficient in complex calculations, they can invoke a calculator tool to handle such tasks. Motivated by this approach, researchers across various fields have developed intelligent task solvers tailored to specific domains. For instance, Wu et al. (2023) proposed integrating ChatGPT with visual foundation models to tackle professional visual tasks. In the GIS domain, Zhang et al. (2024e) introduced GeoGPT, a system designed to solve geospatial tasks by utilizing professional GIS tools within a meticulously curated GIS tool pool. Additionally, this concept has been applied in other areas, such as chemistry (Boiko et al., 2023), public health (Dai et al., 2023), cartography (Zhang et al., 2024c), and remote sensing (Xu et al., 2024b).

However, a significant challenge is that most of these studies rely on closed-source commercial LLMs, such as ChatGPT and GPT-4, which are accessible only via APIs, thus limiting their deployment on local private devices. Recognizing this issue, some researchers have proposed solutions to enhance the tool-use capabilities of open-source LLMs. For instance, Qin et al. (2023) introduced a framework for fine-tuning the open-source LLaMA model to utilize real-world APIs. Similarly, Yang et al. (2024) collected an instruction-following dataset and fine-tuned two open-source LLMs, LLaMA and OPT, to improve their ability to use vision-related tools.

Building on the contributions of previous research, our focus is on enhancing the geospatial tool-use capabilities of open-source LLMs. As discussed in the Introduction, geospatial tasks often require a sequence of multiple tools and steps, with the output of earlier tools frequently serving as the input for subsequent tools, which is referred to as a tool-use chain. The need for such a tool-use chain complicates the process of collecting relevant instruction-tuning data. To address this challenge, we propose a seed task-guided self-instruct strategy for generating a geospatial tool-use instruction-tuning dataset within a simulated environment. Details of our strategy will be elaborated in Sections 3 and 4.

3. Data collection and processing

In this paper, we aim to improve the ability of general-domain open-source LLMs to solve geospatial tasks through instruction tuning. For this purpose, we collected two types of datasets: an Instruction Tuning dataset, GTChain-IT, consisting of geospatial tasks and their corresponding tool-use chains (Section 3.1), and an evaluation dataset, GTChain-Eval, to assess the performance of LLMs in solving geospatial tasks (Section 3.2). In this section, the process of collecting the instruction tuning data and the evaluation data is elaborated.

3.1. Instruction tuning data: GTChain-IT

Instruction tuning is a critical stage in training LLMs, aimed at aligning their responses with the chat styles of humans or experts in specific domains (Ouyang et al., 2022). Many studies have validated the effectiveness of instruction tuning in improving the response quality of LLMs in various fields, such as law (Cui et al., 2023), medicine (Thirunavukarasu et al., 2023), and GIS (Zhang et al., 2024d). Two elements are essential in determining the performance of instruction tuning: the diversity of instructions (i.e., the input questions for LLMs) and the quality of the corresponding responses (i.e., the desired responses from LLMs) (Taori et al., 2023; Wang et al., 2022). In this section, we elaborate on our designed strategy for collecting instruction tuning data, which considers these two elements. Specifically, there are four stages to collect the instruction tuning data GTChain-IT: geospatial tool collection (Section 3.1.1), diverse geospatial task production (Section 3.1.2), tool-use chain generation within a simulated environment (Section 3.1.3), instruction tuning data construction (Section 3.1.4).

3.1.1. Geospatial tool collection

First, a set of geospatial tools with different functions is collected to enable LLMs to solve geospatial tasks. Specifically, 27 commonly used geospatial tools from 8 categories are included in this paper, and all the tools are listed in Table 1. Since LLMs can only accept natural text as input and output, it is important to guide them to understand the function and usage of each tool. In this way, a formal template is designed for each tool t_i in Eq. (1).

$$\begin{aligned} t_i &= \{n_i, f_i, \alpha_i, \beta_i, e_i\} \\ T &= \{t_1, t_2, \ldots, t_{27}\} \end{aligned} \tag{1}$$

where n_i is the name of the tool t_i, f_i is the textural function description, α_i and β_i denote the input and output requirement or format of this tool, respectively, e_i is a manually crafted geospatial task example that can be solved by this tool, and T is the whole tool collection. Table 1 provides tool names and function descriptions of the collected tools, and all the other details can be viewed in Appendix A. Take the tool Buffer as an example (see Fig. 2), all the tool descriptions are clear and concise, which aims to reduce the input text (i.e., token) load for LLMs.

3.1.2. Diverse geospatial task production

As mentioned above, the diversity of instructions is crucial for the performance of instruction tuning. With the collected toolset T, it is also important to produce diverse geospatial tasks within various application scenarios. However, manually designing various geospatial tasks would be labor-intensive. To address this, a seed task-guided self-instruct strategy is designed.

Self-instruct is a recently proposed strategy that aims to alleviate the workload of human annotators with the help of LLMs (Wang et al., 2022). Specifically, training an LLM generally requires a large-scale

Table 1
Collected geospatial tools.

Tool type	Tool name	Tool function
Data download (5)	Get POI By Keywords	Gets urban POI data with a specific category in a city of China online
	Get POI By Rectangle	Gets urban POI data within a rectangle area of China online
	Get Road Network By Rectangle	Gets road network data within a rectangle area online
	Get Remote Sensing Image	Gets remote sensing image data within a time span in a city online
	Obtain One POI	Gets a shapefile containing a specific POI place in China online
Overlay analysis (4)	Union	Computes a geometric union of the input shapefiles
	Intersect	Finds the intersection (overlap) between two shapefiles
	Clip	Extracts objects from shapefile A that are within the extent of shapefile B
	Erase	Removes objects from shapefile A that overlap with shapefile B
Proximity analysis (4)	Buffer	Creates a buffer zone around a shapefile
	Near	Calculates distance and proximity information between objects in an input shapefile and the closest objects in another reference shapefile
	Create Thiessen Polygons	Creates Thiessen polygons from a point shapefile. Each Thiessen polygon contains only a single point. Any location within a Thiessen polygon is closer to its associated point than to any other points
	Point Distance	Determines the distances from the input point shapefile to all points in the near point shapefile
Pattern analysis (3)	Kriging	Interpolates a raster surface from points using kriging
	Kernel Density	Calculates a magnitude-per-unit area from point or polyline shapefile using a kernel function to fit a smoothly tapered surface to each point or polyline
	Point Clustering	Groups points in a point shapefile based on their spatial distance
Attribute analysis (2)	Select	Extracts objects from a shapefile based on a SQL expression
	Frequency	Analyzes a table for unique field values and their frequency of occurrence
Raster processing (4)	Extract Values to Points	Extracts the cell values of a raster based on a point shapefile and records the values in the attribute table of an output point shapefile
	Slope	Identifies the slope (gradient or steepness) from each cell of a raster
	Aspect	Derives the aspect from each cell of a raster surface. The aspect identifies the compass direction that the downhill slope faces for each location
	Clip Raster	Clip input raster data according to the shape of another clip raster or shapefile data
Remote sensing image interpretation (4)	Road Network Extraction From Remote Sensing Image	Extracts road network from input remote sensing image
	Land Use Classification Based on Remote Sensing Image	Categorizes different types of land cover in an area based on input remote sensing data
	Water Extraction From Remote Sensing Image	Identifies water bodies, such as rivers, lakes, reservoirs, and ponds, from input remote sensing image data
	Building Extraction From Remote Sensing Image	Extracts buildings from input remote sensing image
Data output (1)	Visualization	Visualizing a vector dataset as an image

training corpus, and relying solely on humans to label all the training data is a complex and time-consuming task. Given the advanced semantic understanding, reasoning, and text generation abilities of some LLMs (e.g., GPT-4), researchers have attempted to use robust LLMs to assist with text labeling and have obtained effective instruction tuning data. For example, Taori et al. (2023) used GPT-4 to generate 52,000 instruction data via the self-instruct strategy, and the results showed that LLaMA-7B fine-tuned on this dataset could match the performance of text-davinci-003, which has a much larger parameter size.

Therefore, the self-instruct strategy is also adopted to generate instruction tuning data. However, different from general-domain instructions, this paper focuses on generating various geospatial tasks, which require more professional knowledge. Moreover, since the instruction data is collected for improving the geospatial tool-use ability of LLMs, the generated instructions should be related to the collected tools. In this way, relying only on self-instruct prompts to generate geospatial tasks may not always meet these requirements. To address this problem, a seed task (see Fig. 2) is manually crafted for each tool in this paper, which aims to provide the used LLM with more professional information. The LLM can then be guided to generate various geospatial tasks by learning from and simulating the seed tasks. In this paper, GPT-4, one of the most robust LLMs, is adopted for this task.

Specifically, two prompts are designed for single-tool geospatial task production and multiple-tools geospatial task production, respectively. For single-tool geospatial task production (Fig. 3a), GPT-4 is asked to generate instructions that can be solved by one specific tool. Moreover, the instructions should contain all the needed information, such as the provided data path and input parameters. A manually crafted example task of a specific tool is also provided in the prompt to further guide the LLM in understanding the requirements of a geospatial task. For multiple-tools geospatial task production (Fig. 3b), a similar prompt is designed, but it requires combining all the provided tools to generate the geospatial tasks. In other words, the generated geospatial tasks should be solved by using all the provided tools. With these two prompts, geospatial tasks within various application scenarios and different levels of complexity (i.e., the number of tools required to solve them) can be collected.

3.1.3. Tool-use chain generation within a simulated environment

To construct an instruction tuning dataset, obtaining high-quality responses to corresponding instructions is essential. Therefore, with the collected geospatial tasks, it is crucial to generate accurate corresponding tool-use chain solutions. In practice, a direct way to obtain tool-use chain solutions for these geospatial tasks is by inviting GIS professionals to manually solve them using GIS software (e.g., ArcGIS) and recording the processes. However, this method is labor-intensive. Consequently, we also explore an effective strategy to employ a robust LLM (GPT-4 in

As a Geographic Information Science expert, you possess extensive knowledge in utilizing tools for solving geospatial tasks.
Below is one geospatial tool:
{Tool_Name}: {Function} Its input requires {Input}
Your task is to understand the function of this tool and generate a realistic geospatial task case.
Your task description should be clear and concise in no longer than three sentences.
You should generate simulated input parameters (such as file path, field names, or numeric values) the tool need.
Your task description should not contain any words in the tool name or function.
Here is an example of the task that can be solved by this tool, please generate diverse task styles, but follow the format provided in the example:
{Task Example}
Ensure the geospatial task is clear and can be solved solely using this tool.
Do not output the processing step, just task.

(a) The prompt for single-tool geospatial task production

As a Geographic Information Science expert, you possess extensive knowledge in utilizing tools for solving geospatial tasks.
Below is a list of geospatial tools:
Tool 1. {Tool_Name}: {Function} Its input requires {Input}
Tool 2. {Tool_Name}: {Function} Its input requires {Input}
...
Your task is to understand the function of these tools and combine them to generate a realistic geospatial task case.
Your task description should be clear and concise in no longer than six sentences.
You should generate simulated input parameters (such as file paths, field names, or numeric values) the tools need.
Your task description should not contain any words in tool names or functions.
Here are examples of the task that can be solved by a corresponding tool, please generate diverse task styles, but follow the format provided in the examples:
{Task Examples}
Ensure the geospatial task is clear and requires the combination of all the tools for resolution.
Each tool can be used multiple times if needed. Ensure that the output of some tools can be used as the input of other tools.
Do not output the processing step, just task.

(b) The prompt for multiple-tools geospatial task production

Fig. 3. The prompts used for geospatial task production by the seed-task self-instruct strategy.

As a Geographic Information Science expert, you possess extensive knowledge in utilizing tools for solving geospatial tasks.
Below is a list of geospatial tools:
Tool 1. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
Tool 2. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
...
Your task is to understand the function, input, and output of these tools and combine them to solve a realistic geospatial task case.
Task:{**Geosaptial task**}
You need to first assess whether these tools can solve the task.
If they can, you should first output a solution framework to solve this task using the provided tools, and then output the detailed processing steps in the specified format:
Framework: I can solve this task, and I will first use tool_1_name for ..., and then I will use tool_2_name for ..., finally I can obtain the results.
Workflow: [tool_1_name, (arguments_to_tool_1 including any specific parameters), (tool_1_output)]; [tool_2_name, (arguments_to_tool_2 including any specific parameters), (tool_2_output)]; ...
If they cannot, you should output the reason in the specified format:
Framework: I cannot solve this task, and the reason is (such as parameters or data are not clear, existing tools do not have the capacity to solve this problem, ...)
Do not modify the tool name, add fake tools. You should provide the simulated output data path of each tool, and you can also use the output data.
Do not output other content, just the Framework and Workflow strictly following the specified format.

Fig. 4. The prompt used for tool-use chain generation within a simulated environment.

this case) to automatically generate a tool-use chain for each geospatial task.

Specifically, a prompt is designed to guide the LLM to simulate the task-solving process (Fig. 4). First, the information of tools used to generate this task from the last step is provided. Specifically, tool descriptions, including tool names, functions, inputs, and required outputs, are provided in this prompt. This enables the LLM to understand the usage of each tool. Given the collected geospatial tasks from Section 3.1.2 (as illustrated by the bold Geospatial task in Fig. 4), the LLM is required to comprehend the tools and combine them to solve the task. Since the LLM cannot truly solve the task in the same way a GIS professional would operate GIS software, we define an output format to simulate this process. Specifically, the LLM should produce two parts in its output. First, it should assess whether the provided tools can solve the given geospatial task. If these tools can be combined to solve the task, the LLM should first output a framework to clarify the tool-use chain. Additionally, it should produce a workflow to describe the detailed data stream.

Since LLMs may produce content with hallucinations, requiring these two parts serves as a self-check mechanism. The tool names and corresponding sequences in the framework can be matched with those in the workflow, and any mismatch indicates that the LLM's output is inaccurate and should be discarded. Moreover, relevant studies suggest that step-by-step thinking can improve the response quality of LLMs. With this strategy, the tool-use chain of each geospatial task can be obtained.

"**instruction**": "As a Geographic Information Science expert, you possess extensive knowledge in utilizing tools for solving geospatial tasks.
Below is a list of geospatial tools:
Tool 1. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
Tool 2. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
…
Tool 27. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
Your task is to understand the function, input, and output of these tools and combine them to solve a realistic geospatial task case.
You need to first assess whether these tools can solve the task.
If they can, you should first output a solution framework to solve this task using the provided tools, and then output the detailed processing steps in the specified format:
Workflow: [tool_1_name, (arguments_to_tool_1 including any specific parameters), (tool_1_output)]; [tool_2_name, (arguments_to_tool_2 including any specific parameters), (tool_2_output)]; …
If they cannot, you should output the reason in the specified format:
Framework: I cannot solve this task, and the reason is (such as parameters or data are not clear, existing tools do not have the capacity to solve this problem, ...)
Do not modify the tool name, add fake tools. You should provide the simulated output data path of each tool, and you can also use the output data.
Do not output other content, just the Framework and Workflow strictly following the specified format.",

(a) System prompt

"**input**": "I am planning a new residential development project and need to ensure that the proposed sites are not only within 1 km of existing schools but also overlap with designated residential zones. Additionally, the sites should exclude any areas designated as industrial zones to maintain a peaceful living environment. The relevant files are located as follows: schools at ./schools.shp, residential zones at ./residential_zones.shp, and industrial zones at ./industrial_zones.shp. Please use these files to identify suitable locations for the development.",

(b) Geospatial task

"**output**": "Workflow: [Buffer, (./schools.shp, 1000), (./buffered_schools.shp)]; [Intersect, (./buffered_schools.shp, ./residential_zones.shp), (./residential_buffered_schools.shp)]; [Erase, (./residential_buffered_schools.shp, ./industrial_zones.shp), (./suitable_locations.shp)]"

(c) Tool-use chain

Fig. 5. A case example of the instruction tuning data, GTChain-IT, consists of a system prompt, a geospatial task, and a corresponding tool-use chain to solve this task.

3.1.4. Instruction tuning data construction

With the former two steps, geospatial tasks within various application scenarios and their corresponding tool-use chains can be obtained. Therefore, the instruction tuning data GTChain-IT can be constructed. An instruction case is presented in Fig. 5. In this paper, GTChain-IT can be formalized as Eq. (2).

$$IT_i = \{instruction, input_i, output_i\}$$
$$GTChain\text{-}IT = \{IT_1, IT_2, \ldots, IT_n\} \quad (2)$$

where (Fig. 5a) is a system prompt used to guide the trained LLM to know how many tools there are and the usage of each tool. Different from the prompt in Fig. 4, the information of all 27 tools is provided here. $instruction$ is the same in all IT_i. Then, $input_i$ (Fig. 5b) is the collected geospatial task, $instruction$ and $input_i$ together construct the input for the trained LLM, and $output_i$ (Fig. 5c) is the collected tool-use chain for each task, which is also the desired output of the trained LLM. In this paper, the workflow obtained from the last step is used as the desired output.

Moreover, after conducting a literature review, we found that some studies also construct their instruction tuning data in the form presented in Fig. 6 (Yang et al., 2024). Specifically, the workflow in Fig. 5 is reorganized into the iterative "Action, Action Input, Observation" process. To explore the influence of data format, GTChain-IT is also constructed in this format, called GTChain-IT (AAO). The relevant discussion is presented in the experiment section.

In this paper, to generate instruction tuning data with varying levels of complexity, three levels of geospatial tasks and corresponding tool-use chains are collected, including tasks requiring a single tool, two tools, and three tools for resolution. Specifically, for tasks requiring a single tool, each tool is repeated 50 times to enable LLMs to thoroughly understand the function and usage of each tool ($27 \times 50 = 1350$). All possible combinations of two tools are repeated 5 times to help LLMs learn how to solve geospatial tasks using different tool combinations ($27 \times 26 \div 2 \times 5 = 1755$). After the matching process between the framework and workflow, some unmatched cases were removed, and we finally obtained 1700 instructions for two tools. For tasks requiring three tools, the 27 available tools are grouped into sets of three tools each, resulting in 9 groups per iteration. This grouping process is repeated 120 times. Finally, after removing unmatched cases, 950 instructions are retained. The statistics of all the collected instruction tuning data are presented in Table 2. *Tool number* in Table 2 refers to the number of tools required to solve each corresponding geospatial task. Finally, the overall workflow to collect the instruction tuning data is presented in Fig. 7.

Table 2
Statistics of the collected GTChain-IT.

Tool number	1	2	3	Total
Instruction number	1350	1700	950	4000

3.2. Evaluation data: GTChain-eval

To evaluate the ability of LLMs to solve geospatial tasks, evaluation data is also collected. Similar to the instruction tuning data, three levels of geospatial tasks are collected. For geospatial tasks requiring a single tool for resolution, 216 tasks are collected (27 tools, each repeated 8 times). For tasks requiring two tools, the 27 available tools are grouped into sets of two tools each, resulting in 13 unique pairs. This grouping process is repeated 8 times, yielding a total of 104 tasks. Finally, for tasks requiring three tools, the 27 available tools are grouped into sets of three tools each, resulting in 9 unique triplets. This grouping process is repeated 9 times, with the first 80 tasks randomly selected. Different from the tool-use chain generation

"**instruction**": "As a Geographic Information Science expert, you possess extensive knowledge in utilizing tools for solving geospatial tasks.
...
Action: tool_1_name
Action Input: arguments_to_tool_1 including any specific parameters
Observation: tool_1_output
Action: tool_2_name
Action Input: arguments_to_tool_2 including any specific parameters
Observation: tool_2_output ...
...

(a) System prompt

"**output**": "
Action: Buffer
Action Input: ./schools.shp, 1000
Observation: ./buffered_schools.shp
Action: Intersect
Action Input: ./buffered_schools.shp, ./residential_zones.shp
Observation: ./residential_buffered_schools.shp
Action: Erase
Action Input: ./residential_buffered_schools.shp, ./industrial_zones.shp
Observation: ./suitable_locations.shp"

(b) Tool-use chain

Fig. 6. Instruction tuning data in the form of "Action, Action Input, Observation".

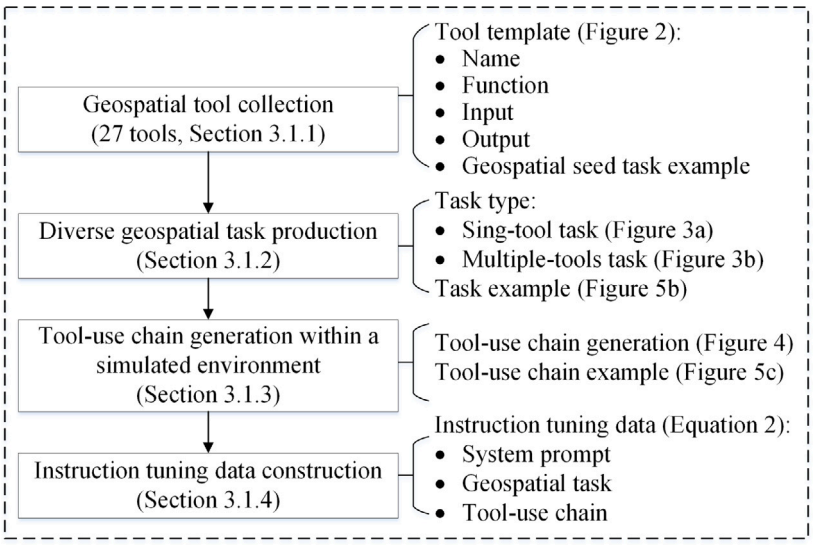


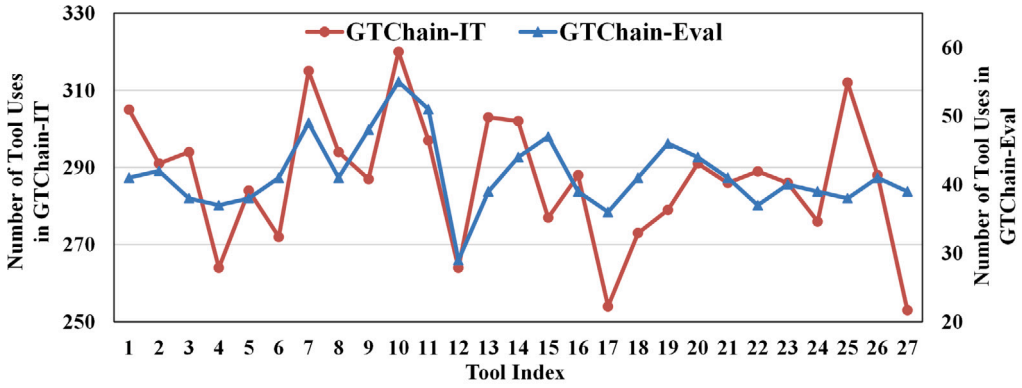
Fig. 7. The workflow to collect instruction tuning data.

process to collect GTChain-IT by only an LLM, the evaluation data is obtained with further professional checks. Specifically, the details of the generated geospatial tasks, and each step of the framework and workflow (see Fig. 5b and c), are checked by two GIS professionals with master's degrees. In this way, the validity of the evaluation data can be guaranteed, ensuring it can be used for effective evaluation. The statistics of all the collected evaluation data are presented in Table 3. Moreover, since some geospatial tasks may require using a tool multiple times for resolution (see Fig. 1), the statistics of the number of uses for each tool in GTChain-IT and GTChain-Eval are also presented in Fig. 8.

Moreover, two extended datasets were also collected to evaluate the performance of GTChain beyond the scenarios in the training data. Specifically, the first extended dataset, GTChain-EvalExtend1, consists of geospatial tasks requiring four and five geospatial tools for resolution. With GTChain-EvalExtend1, GTChain can be tested on more complex tasks than those in the training data. Additionally, five external geospatial tools were collected, and the second extended dataset, GTChain-EvalExtend2, was constructed based on these tools that were not used during the training process of GTChain. The function descriptions of these external five tools are presented in Table 4, more detailed information about them are listed in Appendix B. Specifically, three levels of geospatial tasks were collected. For geospatial tasks requiring one tool for resolution, 20 instructions were collected (5 tools, each repeated 4 times). For geospatial tasks requiring two tools, 15 instructions were collected, with each instruction generated based on

Table 3
Statistics of the collected GTChain-Eval, GTChain-EvalExtend1, and GTChain-EvalExtend2.

Tool number	1	2	3	4	5	Total
GTChain-Eval	216	104	80	–	–	400
GTChain-EvalExtend1	–	–	–	30	20	50
GTChain-EvalExtend2	20	15	15	–	–	50



1. Aspect	10. Get POI By Keywords	19. Obtain one POI
2. Buffer	11. Get POI By Rectangle	20. Point Clustering
3. Building Extraction From Remote Sensing Image	12. Get Remote Sensing Image	21. Point Distance
4. Clip Raster	13. Get Road Network By Rectangle	22. Road Network Extraction From Remote Sensing Image
5. Clip	14. Intersect	23. Select
6. Create Thiessen polygons	15. Kernel Density	24. Slope
7. Erase	16. Kriging	25. Union
8. Extract Values to Points	17. Land Use Classification Based on Remote Sensing Image	26. Visualization
9. Frequency	18. Near	27. Water Extraction From Remote Sensing Image

Fig. 8. Statistics of the number of uses for each tool in GTChain-IT and GTChain-Eval.

Table 4
The details of five external geospatial tools.

Tool name	Tool function
NDVI	Calculates the Normalized Difference Vegetation Index (NDVI) from the raster data
Resample	Resample a raster file to a different resolution or grid size
Summary Statistics	Computes summary statistics such as mean, median, mode, and standard deviation for the specified data field within a dataset
Raster2Vector	Converts raster data to vector format
Mosaic	Merges multiple existing raster datasets into an existing raster dataset

one external tool and a tool from the 27 training tools. Similarly, 15 instructions were collected for geospatial tasks requiring three tools for resolution. In this way, with GTChain-EvalExtend2, GTChain can be tested on unfamiliar geospatial tools to demonstrate its transfer performance.

4. The training process

4.1. Parameter efficient fine tuning

In this section, the process of training GTChain is described. Generally, fine-tuning is one of the most effective strategies for adapting a pretrained general model to specific tasks. Unlike training a model from scratch, fine-tuning uses task-specific data to further train a general model, improving its performance on specific tasks while saving data volume and computing resources. However, for LLMs, their large number of parameters (billions) makes it unfeasible to fine-tune the entire network without sufficient data and computing resources.

To address this, researchers have developed Parameter Efficient Fine Tuning (PEFT) strategies for large general models and LLMs. With different considerations, several effective PEFT methods have been proposed, including LoRA (Low-Rank Adaptation) (Hu et al., 2021), adapters (Houlsby et al., 2019), prompt tuning (Lester et al., 2021), and prefix-tuning (Li and Liang, 2021), to name a few. Among these methods, due to the unique strategy of introducing low-rank matrices into pre-trained models, LoRA enables significant reduction in the number of tunable parameters without altering the model's core weights, making it highly suitable for large-scale models. By contrast, adapters add layers within the model, which can improve task performance but increase computational and storage demands. Prompt tuning and prefix-tuning both modify only the input prompts rather than internal weights, making them lightweight yet limiting their effectiveness in tasks requiring deeper model adaptation. While prompt tuning and prefix-tuning excel in specific or few-shot tasks, LoRA offers a more versatile and robust solution for complex scenarios, balancing efficiency with strong performance across a broader range of applications. This makes LoRA particularly effective for scenarios requiring flexible deployment and memory efficiency without sacrificing model adaptability. Moreover, the effectiveness of LoRA for training professional LLMs has been validated in various domains such as law (Wang et al., 2024a), medicine (Liu et al., 2024), and GIS (Zhang et al., 2024d). Therefore, LoRA is also adopted in this paper.

Specifically, LoRA is adopted to further fine-tune GTChain based on the top of a general-domain LLM, LLaMA-2-7B. LLaMA-2-7B is an open-sourced foundational LLM proposed by Meta, which has shown the potential to be fine-tuned in professional domains. Therefore, with the collected instruction tuning data, LLaMA-2-7B is fine-tuned to maximally align the labeled tool-use chains given input geospatial tasks based on LoRA. The process of LoRA is described in Eq. (3).

$$h = Wx + \Delta Wx = Wx + BAx \tag{3}$$

where x is an input d dimensional feature, $W \in R^{d\times k}$ is the original network parameter, h is the output feature after this network. For LLMs, fine-tuning the parameters of W is computing-intensive. In this way, another sub-channel network is designed, that is $\Delta W = BAx$, where $B \in R^{d\times r}$, $A \in R^{r\times k}$, and $r \ll min(d, k)$. During the training process, the parameters of W are fixed, which do not require gradient calculation. Meanwhile, the parameters of ΔW (i.e., A and B) will be tuned to adapt to our focused task, which requires much fewer computing resources.

4.2. Training setting

To obtain GTChain, the hyperparameters setting of this fine-tuning process are as follows: the LoRA rank is set as 8, and the modules k_proj, q_proj, v_proj, o_proj, and MLP layers in LLaMA-2-7B is set as the focused modules (i.e., w). During the training process, 8 bit quantization is used on all the input features for saving computing resources. The batch size is set as 1, the learning rate is set as 1e-4, and the maximum target length of the model is set as 4096 tokens. The model is fine-tuned for 1 epoch on 2×NVIDIA GeForce RTX 4090 GPU.

4.3. Workflow of model training

Finally, to clearly present the workflow of model training, a pseudocode framework is provided in Algorithm 1. During model training, the input data is the collected instruction tuning data, GTChain-IT. The models consist of the original LLaMA-2-7B and LoRA modules. During the training process, the original LLaMA-2-7B is frozen, and only the designed LoRA modules are trainable to save computing resources. During each iteration, the original LLaMA-2-7B model first embeds the input $instruction$ and $input_i$ as a hidden feature x. Then, at the LoRA module stage, the hidden feature is processed by both the original layers of LLaMA-2-7B (i.e., k_proj, q_proj, v_proj, o_proj, and MLP layers) and the designed LoRA module. Finally, at the final layer, the obtained hidden feature h is used to predict the result y, and the corresponding loss is calculated based on the label $output_i$, which is used to update the weights of all LoRA modules.

Algorithm 1: The workflow for training GTChain

Data: The collected instruction tuning data
$GTChain\text{-}IT = \{IT_i\}_{i=1}^{n}$; $IT_i = \{instruction, input_i, output_i\}$;
// see Eq. (2)
Model: Original LLaMA-2-7B (frozen); LoRA modules (trainable);
for $epoch \leftarrow 1$ **to** 1 **do**
  for $i \leftarrow 1$ **to** n **do**
    // input $instruction$ and $input_i$ to the original model and obtain the hidden feature x
    ...
    $h = Wx + \Delta Wx = Wx + BAx$ // see Eq. (3)
    ...
    // obtain the predicted result y based on hidden feature h
    $loss = L(y, output_i)$ // L denotes the Cross-Entropy Loss
    // the obtained loss will be used to update ΔW

5. Evaluation

In this section, the evaluation process is introduced, including baseline (Section 5.1) and results (Section 5.2). The ablation study, including the impact of the instruction tuning data format and the impact of the "framework-workflow" structure, is discussed in Section 5.3. Moreover, to demonstrate the geospatial task-solving process by GTChain, some case studies are presented in Section 5.6.

5.1. Baseline

First, to validate the effectiveness of our collected instruction tuning data and the training strategy used, GTChain is compared with the original LLaMA-2-7B to show the improvement our model can achieve. Second, GTChain is also compared with the general-domain open-source LLMs Vicuna (Version 1.5) and Mistral-7B, which has a similar parameter size to GTChain. Moreover, GTChain is compared with robust, commercially-used LLMs, including GPT-4, GPT-3.5-turbo, and Gemini 1.5 Pro, to determine if there is a performance gap between our model and the state-of-the-art models. During the evaluation process, the system prompt $instruction$ and each geospatial task $input_i$ will serve as the input for baselines and GTChain. Their output will be manually evaluated by comparing the corresponding $output_i$ to calculate their accuracy. Note that accuracy is defined as the percentage of correct outputs (i.e., tool-use chains) generated by corresponding models.

5.2. Result

First, the evaluation results obtained by the baselines and GTChain on GTChain-Eval are presented in Table 5. To provide fine-scale evaluation results, the number of steps required for each geospatial task is also counted. From the results, GTChain outperforms all other baselines in most cases, including three of the most robust commercial models: GPT-3.5-turbo, GPT-4, and Gemini 1.5 Pro. Notably, GTChain is trained on top of LLaMA-2-7B, which has a much smaller parameter size compared to these models. Therefore, the results validate the effectiveness of our proposed framework. Additionally, the results indicate that Gemini 1.5 Pro performs similarly to, but slightly better than, GPT-4 on GTChain-Eval, with both models significantly outperforming GPT-3.5-turbo. This improvement can be attributed to the fact that both Gemini 1.5 Pro and GPT-4 have been specifically trained for tool usage. In contrast, while Vicuna demonstrates robust semantic understanding in general domains, its performance on professional GIS tasks is less satisfactory. Vicuna is also trained on LLaMA-2-7B, which has the same parameter size as GTChain. Thus, the suboptimal performance of Vicuna underscores the importance of collecting specialized training data for models designed to handle specialized tasks. This further supports the effectiveness of the data collected in this study.

Second, the performance of the baselines and GTChain is compared on GTChain-EvalExtend1, which includes tasks requiring more tools (4 and 5 tools) for resolution. During training, GTChain was only trained to use up to 3 tools. Generally, tasks requiring more tools are considered more challenging. According to the results presented in Table 6, GTChain still outperforms all the baselines, indicating that GTChain can handle geospatial tasks of greater complexity than those encountered during its training.

Finally, to further test whether GTChain can utilize external tools that were not encountered during the training process, GTChain is evaluated on the GTChain-EvalExtend2 dataset, with all results presented in Table 7. The results demonstrate that GTChain still outperforms all the baselines. Furthermore, GTChain shows similar accuracy performance on GTChain-EvalExtend2 compared to the results on GTChain-Eval (see Table 5). These findings suggest that GTChain has the capability to adapt and use geospatial tools it has not previously encountered. In other words, with the proposed framework, GTChain can learn to understand and effectively utilize the capabilities of new geospatial tools to address geospatial tasks.

5.3. Ablation study

5.3.1. The impact of the instruction tuning data format

In Section 3.1.4, GTChain-IT (AAO) is introduced, which means the output is organized in the "Action, Action Input, Observation" form. In this section, the influence of the instruction tuning data format is investigated. Specifically, a new model GTChian (AAO) is trained on the top of LLaMA-2-7B based on GTChain-IT (AAO). GTChain (AAO) is also evaluated based on the dataset GTChian-Eval, and all the results are listed in Table 8. From the results, there are only slight differences between the performance of GTChain (AAO) and GTChain, which suggests the format of the instruction tuning dataset may have a slight influence on the model performance on geospatial tool usage.

Table 5
The results obtained by baselines and GTChain on the evaluation dataset GTChain-Eval.

Tool number	Step number	Instruction number	LLaMA-2-7B	Vicuna	Mistral-7B	GPT-3.5-turbo	GPT-4	Gemini 1.5 Pro	GTChain
1	1	135	0.7% (1)	16.3% (22)	3.7% (5)	14.1% (19)	57.8% (78)	63.0% (85)	**96.3% (130)**
1	2	81	3.7% (3)	8.6% (7)	4.9% (4)	28.4% (23)	46.9% (38)	60.5% (49)	**67.9% (55)**
2	2	48	0.0% (0)	0.0% (0)	4.2% (2)	16.7% (8)	31.3% (15)	33.3% (16)	**75.0% (36)**
2	3	32	6.3% (2)	0.0% (0)	0.0% (0)	12.5% (4)	31.3% (10)	31.3% (10)	**78.1% (25)**
2	4	16	0.0% (0)	0.0% (0)	0.0% (0)	25.0% (4)	43.8% (7)	43.8% (7)	**62.5% (10)**
2	5	8	0.0% (0)	0.0% (0)	0.0% (0)	12.5% (1)	25.0% (2)	**37.5% (3)**	**37.5% (3)**
3	3	40	2.5% (1)	0.0% (0)	2.5% (1)	10.0% (4)	20.0% (8)	20.0% (8)	**70.0% (28)**
3	4	20	0.0% (0)	0.0% (0)	0.0% (0)	10.0% (2)	20.0% (4)	5.0% (1)	**25.0% (5)**
3	5	12	0.0% (0)	0.0% (0)	0.0% (0)	25.0% (3)	33.3% (4)	**50.0% (6)**	**50.0% (6)**
3	6	8	0.0% (0)	0.0% (0)	0.0% (0)	37.5% (3)	50.0% (4)	**62.5% (5)**	25.0% (2)
Total		400	1.8% (7)	7.3% (29)	3.0% (12)	17.8% (71)	42.5% (170)	47.5% (190)	**75.0% (300)**

Table 6
The results obtained by baselines and GTChain on the evaluation dataset GTChain-EvalExtend1 (more tools).

Tool number	Step number	Instruction number	LLaMA-2-7B	Vicuna	Mistral-7B	GPT-3.5-turbo	GPT-4	Gemini 1.5 Pro	GTChain
4	4	30	0.0% (0)	0.0% (0)	3.3% (1)	23.3% (7)	33.3% (10)	33.3% (10)	**60.0% (18)**
5	5	20	0.0% (0)	0.0% (0)	10% (2)	20.0% (4)	**35.0% (7)**	30.0% (6)	**35.0% (7)**

Table 7
The results obtained by baselines and GTChain on the evaluation dataset GTChain-EvalExtend2 (external tools).

Tool number	Step number	Instruction number	LLaMA-2-7B	Vicuna	Mistral-7B	GPT-3.5-turbo	GPT-4	Gemini 1.5 Pro	GTChain
1	1	20	10.0% (2)	20.0% (4)	25.0% (5)	35.0% (7)	40.0% (8)	45.0% (9)	**80.0% (16)**
2	2	15	6.7% (1)	6.7% (1)	0.0% (0)	40.0% (6)	40.0% (6)	53.3% (8)	**73.3% (11)**
3	3	15	20.0% (3)	13.3% (2)	20.0% (3)	20.0% (3)	53.3% (8)	66.7% (10)	**73.3% (11)**

Table 8
The results obtained by implementing the model GTChain (AAO) on the dataset GTChain-Eval.

Tool number	Step number	Instruction number	GTChain (AAO)	GTChain
1	1	135	96.3% (130)	96.3% (130)
1	2	81	67.9% (55)	67.9% (55)
2	2	48	79.2% (38)	75.0% (36)
2	3	32	71.9% (23)	78.1% (25)
2	4	16	43.8% (7)	62.5% (10)
2	5	8	37.5% (3)	37.5% (3)
3	3	40	70.0% (28)	70.0% (28)
3	4	20	25.0% (5)	25.0% (5)
3	5	12	41.7% (5)	50.0% (6)
3	6	8	25.0% (2)	25.0% (2)

Table 9
The results obtained by implementing the model GTChain (FW) on the dataset GTChain-Eval.

Tool number	Step number	Instruction number	GTChain (FW)	GTChain
1	1	135	90.4% (122)	96.3% (130)
1	2	81	58.0% (47)	67.9% (55)
2	2	48	83.3% (40)	75.0% (36)
2	3	32	46.9% (15)	78.1% (25)
2	4	16	25.0% (4)	62.5% (10)
2	5	8	50.0% (4)	37.5% (3)
3	3	40	60.0% (24)	70.0% (28)
3	4	20	25.0% (5)	25.0% (5)
3	5	12	16.7% (2)	50.0% (6)
3	6	8	37.5% (3)	25.0% (2)

5.3.2. The impact of the "framework-workflow" structure

During the collection of GTChain-IT, a two-step strategy is adopted. Specifically, when generating a solution for a geospatial task, the used robust LLM (GPT-4) is guided to first output a framework about how it will solve this task (or whether it can solve this task), and then output the workflow of the detailed data stream. Therefore, a question is what is the impact if the instruction tuning data is organized as the "framework-workflow" structure like that in Fig. 9c. In this section, all the instruction tuning data are organized in the "framework-workflow", termed GTChain-IT (FW), and a new model called GTChain (FW) is trained on the top of LLaMA-2-7B based on GTChain-IT (FW). Finally, the GTChain (FW) is evaluated on the dataset GTChian-Eval, and all the results are listed in Table 9. From the results, GTChain (FW) demonstrates poorer performance than GTChain in most cases. One potential reason is that when the instruction-tuning data is reorganized into longer text (i.e., GTChain-IT (FW)), it increases the model's burden in inferring accurate results. Therefore, the requirement for concise output may be better suited to our model.

5.3.3. The generalization ability across different types of geospatial data

In this section, we investigate the generalization ability of GTChain across different types of geospatial data. Specifically, we count the number of instructions related to various data types, including point shapefiles, line shapefiles, polygon shapefiles, raster data, and remote sensing imagery. Note that some instructions may be relevant to multiple data types, so the total count of these instructions is higher than the count of GTChain-Eval. The results are presented in Table 10. Based on the results, GTChain shows relatively slight differences in accuracy when processing different data types, validating the generalization of our framework.

Table 10
The experimental results of GTChain across different types of geospatial data.

Data type	Instruction number	Accuracy (%)
Point shapefile	207	72.0% (149)
Line shapefile	77	79.2% (61)
Polygon shapefile	138	65.2% (90)
Raster data	125	70.4% (88)
Remote sensing imagery	102	73.5% (75)

5.3.4. The impact of LoRA rank r

In this section, we investigate the impact of LoRA rank r on model performance. In practice, the value of the LoRA rank r determines the size of the LoRA modules during training; specifically, a higher rank corresponds to larger LoRA module sizes. We evaluate GTChain

"**instruction**": "As a Geographic Information Science expert, you possess extensive knowledge in utilizing tools for solving geospatial tasks.
Below is a list of geospatial tools:
Tool 1. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
Tool 2. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
…
Tool 27. {Tool_Name}: {Function} Its input requires {Input} Its output is {Output}
Your task is to understand the function, input, and output of these tools and combine them to solve a realistic geospatial task case.
You need to first assess whether these tools can solve the task.
If they can, you should first output a solution framework to solve this task using the provided tools, and then output the detailed processing steps in the specified format:
Framework: I can solve this task, and I will first use tool_1_name for ..., and then I will use tool_2_name for ..., finally I can obtain the results.
Workflow: [tool_1_name, (arguments_to_tool_1 including any specific parameters), (tool_1_output)]; [tool_2_name, (arguments_to_tool_2 including any specific parameters), (tool_2_output)]; …
If they cannot, you should output the reason in the specified format:
Framework: I cannot solve this task, and the reason is (such as parameters or data are not clear, existing tools do not have the capacity to solve this problem, ...)
Do not modify the tool name, add fake tools. You should provide the simulated output data path of each tool, and you can also use the output data.
Do not output other content, just the Framework and Workflow strictly following the specified format.",

(a) System prompt

"**input**": "I am planning a new residential development project and need to ensure that the proposed sites are not only within 1 km of existing schools but also overlap with designated residential zones. Additionally, the sites should exclude any areas designated as industrial zones to maintain a peaceful living environment. The relevant files are located as follows: schools at ./schools.shp, residential zones at ./residential_zones.shp, and industrial zones at ./industrial_zones.shp. Please use these files to identify suitable locations for the development.",

(b) Geospatial task

"**output**": "Framework: I can solve this task, and I will first use Buffer for creating a 1 km buffer around schools, and then I will use Intersect for finding the overlap between the buffered schools and residential zones, finally, I will use Erase to exclude industrial zones and obtain the results.
Workflow: [Buffer, (./schools.shp, 1000), (./buffered_schools.shp)]; [Intersect, (./buffered_schools.shp, ./residential_zones.shp), (./residential_buffered_schools.shp)]; [Erase, (./residential_buffered_schools.shp, ./industrial_zones.shp), (./suitable_locations.shp)]"

(c) Tool-use chain

Fig. 9. Instruction tuning data with the "framework-workflow" structure.

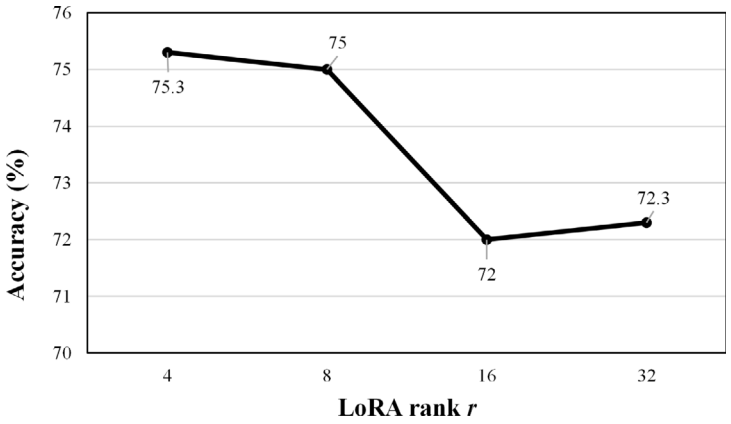


Fig. 10. Performance of GTChain with different LoRA ranks on GTChain-Eval.

with different values of r to examine how LoRA module size influences performance. The results are presented in Fig. 10. As shown, GTChain exhibits varying performance across different values of r, with all settings outperforming the baselines (see Table 5), demonstrating the robustness of our framework. Furthermore, GTChain achieves the best results with a LoRA rank of 4. This suggests that a larger LoRA module size does not necessarily yield better performance, and a moderately balanced rank (e.g., 4) may be optimal for downstream tasks.

5.3.5. The impact of the size of training data GTChain-IT

In this section, we investigate the impact of the size of the training dataset GTChain-IT. Specifically, GTChain is trained with varying proportions of GTChain-IT (100%, 75%, 50%, and 25%) to examine the correlation between training data size and model performance. The results are presented in Fig. 11. From the results, GTChain achieves better performance with more training data, which validates the effectiveness of our data collection strategy. Furthermore, even with only 25% of the training data, GTChain still outperforms the baselines (see Table 5), demonstrating the robustness of our framework.

5.4. Robustness

In practical applications, input data may contain errors or anomalies. Therefore, in this section, the robustness of GTChain to deal with input with anomalies is investigated. Specifically, for each geospatial task in GTChain-Eval, we randomly select words from the input and

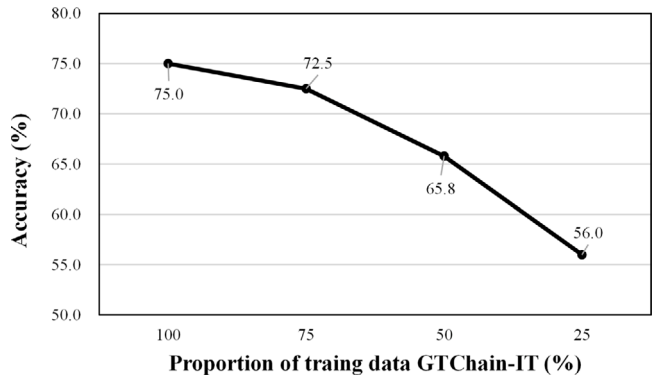


Fig. 11. Performance of GTChain with different proportions of training data on GTChain-Eval.

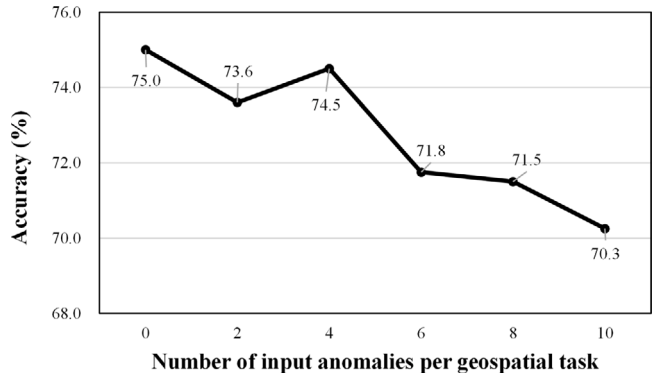


Fig. 12. Performance of GTChain on GTChain-Eval in the presence of input anomalies.

randomly replace characters in the correct words. For example, if the input is 'word', we randomly choose a character in the word (e.g., 'o') and replace it with another character (e.g., 'a'), resulting in the input 'ward'. We conducted multiple experiments to evaluate GTChain's performance under different levels of input anomalies, including cases where each task contains 2, 4, 6, 8, or 10 character errors. All the results are presented in Fig. 12. The results indicate that as the number of input anomalies in the input increases, GTChain's accuracy gradually decreases, which aligns with our general expectations. However, despite the increased errors in the input, GTChain still maintains better accuracy than the baselines. This effectively demonstrates the robustness of our model.

5.5. Efficiency

For many geospatial applications, real-time processing is a key requirement. In this section, we evaluate the computational efficiency of GTChain. Specifically, we test GTChain's efficiency on the evaluation dataset GTChain-Eval and record the average processing time required for tasks with different levels of complexity. The results are presented in Table 11. As shown in the table, the time GTChain takes to solve a task increases with task complexity, which is determined by the number of tools needed for each task. Although GTChain is slightly less time-efficient than GPT-4, its performance remains within an acceptable range.

5.6. Case studies

In this section, two realistic cases in Figs. 13 and 14 are presented to demonstrate how GTChain can assist users in solving geospatial tasks. In the first case, a geospatial task (i.e., the instruction) is proposed,

Table 11
Average processing time of GTChain and GPT-4 for geospatial tasks in GTChain-Eval (in seconds).

Tool number	GTChain	GPT-4
1	7.609	4.655
2	13.125	5.692
3	16.504	7.563
4	17.828	10.358
5	21.232	12.247

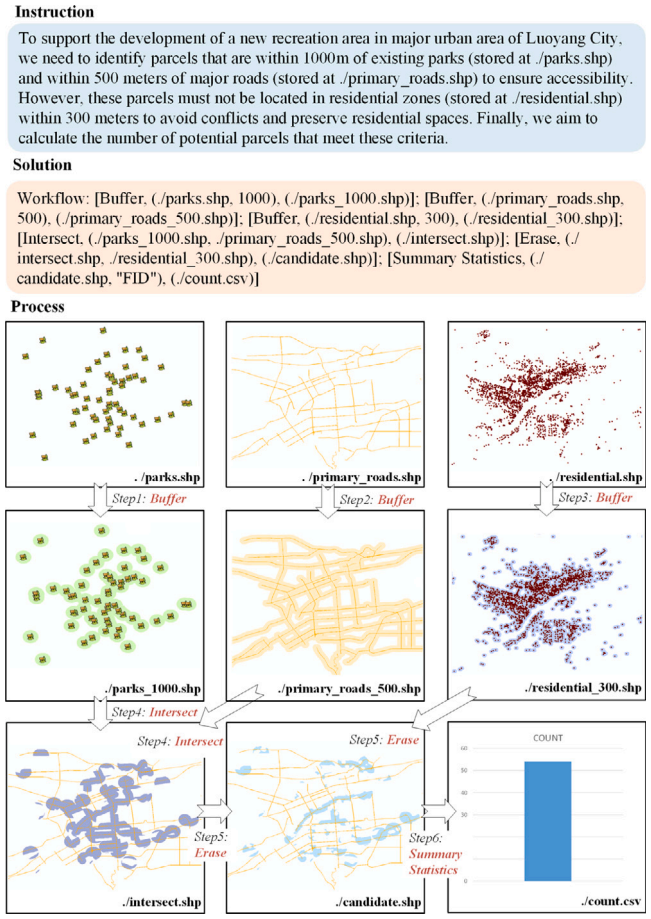


Fig. 13. Case 1: Demonstration of how GTChain solves a geospatial task using four tools (Buffer, Intersect, Erase, and Summary Statistics) across six steps.

and GTChain then accepts this task, generating a workflow to solve it using the knowledge inherent in its model. This workflow includes the tool names and corresponding parameters for each step. Typically, the input parameters for each tool are derived from the instruction or the output of previous steps, while the output data paths are generated by GTChain, which are simulated data paths rather than actual data. Once the workflow is generated, a workflow-tool matching process is automatically conducted. This matching process is straightforward, as the workflow has already specified the entire data stream. Afterward, the corresponding tools are executed, and the task is completed. A similar process is used in case 2.

6. Discussion

6.1. Broader implications

In this paper, we propose a framework to train a general-domain LLM capable of generating correct tool-use chains for specific geospatial tasks. With this framework, users can describe their requirements in

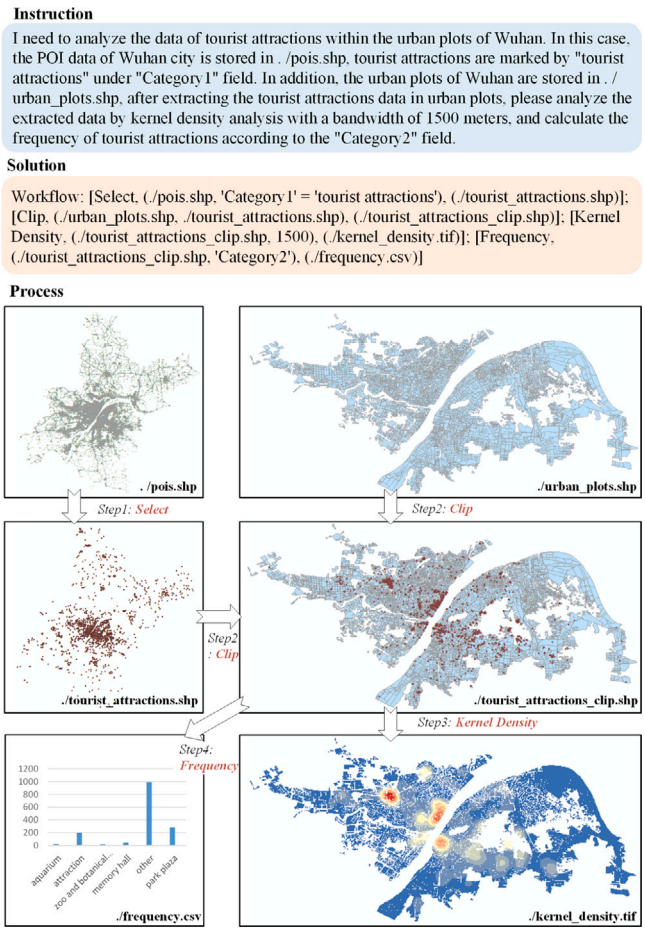


Fig. 14. Case 2: Demonstration of how GTChain solves a geospatial task using four tools (Select, Clip, Kernel Density, and Frequency) across four steps.

natural language, and our model will automatically handle the subsequent steps (as demonstrated in the case studies). This approach transforms the user interface for solving geospatial tasks. Since our framework eliminates the need for users to operate tools themselves, GTChain has the potential to serve as an intelligent assistant for both professional and non-professional users. For instance, Gao and Goodchild (2013) note that the complex hierarchical organization of current desktop-GIS software often requires specific training in GIS skills, creating significant barriers for users. GTChain could address this issue by allowing users to express their needs in natural language. In this way, GTChain has the potential to be an initial attempt at a 'next-generation GIS', which may also promote the adoption of GIS by non-professional users with spatial analysis needs (Li and Ning, 2023; Zhu et al., 2021).

6.2. Limitations

However, there are also limitations and challenges to further applying GTChain in real-world applications, including:

- **Computering requirement**: In this paper, GTChain is built on top of the general-domain LLM LLaMA-2-7B, which has significantly fewer parameters than other commercial LLMs (e.g., GPT-3.5 with 175B parameters). However, it still requires substantial computational resources to operate, which limits its usability on personal computers without high-powered GPUs (e.g., NVIDIA GeForce RTX 4090). Therefore, it is essential to explore effective strategies to reduce the development and deployment cost of GTChain. We plan to address this challenge through two approaches: (1) Model Quantization (Polino et al., 2018): Model quantization is a feasible way to decrease model size and reduce computational requirements. However, balancing model performance with model size is also a challenge. As such, our future work will focus on identifying an effective quantization strategy that maintains model performance while reducing resource demands. (2) Distillation Learning (Zhang et al., 2024b): Distillation learning is another potential method to decrease model size. Specifically, GTChain can serve as a teacher model to train a smaller model to generate tool-use chains for geospatial tasks. In this way, GTChain can eventually be replaced by this smaller model. As with quantization, maintaining a balance between model size and performance is critical. Therefore, developing an effective distillation learning strategy will also be a focus of our future research.
- **Accessible to the general public**: GTChain provides a user-friendly interaction method for solving geospatial tasks by allowing users to input natural language descriptions of their needs. However, although GTChain has the potential to be used by users without professional GIS skills, there is still a gap in making it accessible to the general public. First, it still requires a certain level of professional knowledge for users to accurately describe their requirements. Specifically, if a user has no GIS knowledge, understanding how to articulate their actual needs becomes a challenge, as accurately describing the task is essential for GTChain to produce the correct solution. To address this issue, we plan to train an LLM that can act as a "translator" for non-professional users. This model would engage in conversation with non-professional users, helping them communicate their needs to GTChain using non-technical language. Second, another key challenge is ensuring the robustness of results. Although GTChain outperforms baselines in solving geospatial tasks accurately, it cannot solve all tasks, particularly more complex ones. Therefore, it is important to guide users in understanding whether the results are correct. In future work, we plan to develop a self-check model that can assess the output of GTChain based on the user's requirements.
- **Interpretability**: LLMs are well-known for their robust abilities in semantic understanding and text generation. Many experiments suggest that their excellent performance benefits from both the large size of the models and the vast scale of training data (Kaplan et al., 2020). However, despite their robust capabilities, LLMs are often considered black-box models. Since our focus is on the GIS domain investigating its interpretability is important for advancing relevant research in this area. We plan to explore the interpretability of GTChain from two aspects: (1) Model design: Model architecture is generally critical for performance. We plan to use techniques such as causal tracing (Xu et al., 2024a) to identify the core modules related to GIS knowledge in GTChain. Practically, these insights could also inform model simplification (e.g., by removing non-essential modules). (2) Data: Training data is equally important in the learning process of LLMs. We also intend to explore the interpretability of data by examining which data formats are most beneficial for training the model effectively.

7. Conclusion

In this paper, we propose a framework to enable the open-source LLM LLaMA-2-7B to understand and solve geospatial tasks by teaching it how to use geospatial tools. To achieve this, we propose collecting a geospatial tool-use instruction tuning dataset, GTChain-IT, which can be used to fine-tune the LLM. During the data collection process, a seed task-guided self-instruct strategy is designed to generate geospatial tool-use instructions and corresponding solutions, and a simulated

environment is used to provide simulated geospatial data streams in them. Based on the collected dataset, we employ Low-Rank Adaptation (LoRA) fine-tuning technology to enhance the tool-usage capability of the LLM, enabling it to solve various geospatial tasks. Moreover, we propose a benchmark, GTChain-Eval, to assess whether a LLM can use geospatial tools, whether it can use a greater number of tools than those encountered during training, and whether it can utilize external tools not seen during training. In this benchmark, our fine-tuned GTChain can outperform all the baselines in most cases, including robust commercial LLMs, such as GPT-4 and Gemini 1.5 Pro, which validate the effectiveness of our framework.

To sum up, our contributions are as follows: (1) We introduce a novel concept for automatically solving geospatial tasks based solely on natural language descriptions, providing a user-friendly interaction model for geospatial task-solving. (2) Building upon this concept, we develop a framework to enhance the ability of a general-domain LLM to generate geospatial tool-use chains based on specific task descriptions. (3) Within this framework, we compile a comprehensive dataset, including training data (GTChain-IT) and evaluation data (GTChain-Eval), which can further promote research in this field. (4) To collect this dataset, we introduce a seed task-guided self-instruct strategy that leverages the robust semantic understanding and text generation abilities of LLMs, significantly reducing human resource requirements. Our strategy can also serve as a reference for related studies.

However, the evaluation results show that although GTChain outperforms all the baselines in most cases, it tends to have lower accuracy on more complex geospatial tasks. For example, GTChain performs worse on geospatial tasks requiring 3 tools for resolution compared to tasks requiring only 1 tool. Therefore, improving the robustness of GTChain on more complex geospatial tasks will be a key focus in our future work. Additionally, despite GTChain having a relatively smaller parameter size than commercial models such as GPT-4 and Gemini 1.5 Pro, optimizing its computational resource requirements while maintaining performance is also important. In our future work, we plan to employ techniques such as distillation learning and quantization to optimize its parameter size, making GTChain feasible for use on personal computers. Moreover, we aim to further improve GTChain in three aspects including computational requirements, accessibility, and interpretability. Detailed discussions and potential solutions are provided in Section 6.

CRediT authorship contribution statement

Yifan Zhang: Writing – original draft, Software, Methodology, Investigation, Funding acquisition, Formal analysis, Data curation. **Jingxuan Li:** Supervision, Project administration, Visualization, Software, Methodology, Investigation, Formal analysis, Data curation. **Zhiyun Wang:** Data curation. **Zhengting He:** Data curation. **Qingfeng Guan:** Supervision, Project administration. **Jianfeng Lin:** Supervision, Project administration. **Wenhao Yu:** Writing – review & editing, Supervision, Project administration, Funding acquisition, Formal analysis, Conceptualization.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

The project was supported by Natural Science Foundation of Hubei Province, China (2024AFD412), National Natural Science Foundation of China (42371446 and 42071442), and by the Fundamental Research Funds for National Universities, China University of Geosciences (No. 2024XLA17). This research was also supported by Meituan. Yifan Zhang acknowledges the support of the China Scholarship Council program (Project ID: 202306410085).

Appendix A. Details of the tool set T (27 tools)

Details about the tools can be found in this link https://github.com/AGI-GIS/GTChain .

Appendix B. Details of the external tools (5 tools)

Details about the external tools can be found in this link https://github.com/AGI-GIS/GTChain.

Appendix C. List of abbreviations

- LLM: Large Language Model
- API: Application Programming Interface
- GTChain: The name of our model
- GIS: Geographic Information Systems
- NLP: Natural Language Processing
- GTChain-IT: Our collected instruction tuning dataset
- GTChain-Eval: Our collected benchmark dataset
- GTChain-IT (AAO): GTChain-IT with the format of 'Action, Action Input, Observation'
- GTChain-EvalExtend1: Our collected benchmark dataset with more complex tasks than those in the training data
- GTChain-EvalExtend2: Our collected benchmark dataset with tools that were not used during the training process of GTChain
- PEFT: Parameter Efficient Fine Tuning
- LoRA: Low-Rank Adaptation
- GTChain (AAO): The model trained with the collected data GTChain-IT (AAO)
- GTChain-IT (FW): GTChain-IT with the format of 'framework-workflow'
- GTChain (FW): The model trained with the collected data GTChain-IT (FW)

Data availability

The data and code supporting the findings of this study are available at this link https://github.com/AGI-GIS/GTChain.

References

Abdella, Y., Alfredsen, K., 2010. A gis toolset for automated processing and analysis of radar precipitation data. Comput. Geosci. 36 (4), 422–429.
Albert, L., Rottensteiner, F., Heipke, C., 2014. Land use classification using conditional random fields for the verification of geospatial databases. ISPRS Ann. Photogramm. Remote Sens. Spatial Inf. Sci. 2, 1–7.
Allen, D.W., 2011. Getting to Know Arcgis Modelbuilder. Esri Press.
Boiko, D.A., et al., 2023. Autonomous chemical research with large language models. Nature 624 (7992), 570–578.
Chen, M., et al., 2024. Collaboration between artificial intelligence and earth science communities for mutual benefit. Nat. Geosci. 17 (10), 949–952.
Chu, D., et al., 2024. A multi-view ensemble machine learning approach for 3d modeling using geological and geophysical data. Int. J. Geogr. Inf. Sci. 1–28.
Clerici, A., et al., 2006. A gis-based automated procedure for landslide susceptibility mapping by the conditional analysis method: the Baganza Valley case study (italian northern apennines). Environ. Geol. 50, 941–961.
Cui, J., et al., 2023. Chatlaw: Open-source legal large language model with integrated external knowledge bases. arXiv preprint arXiv:2306.16092.
Dai, H., et al., 2023. Ad-autogpt: An autonomous gpt for Alzheimer's disease infodemiology. arXiv preprint arXiv:2306.10095.
Deng, C., et al., 2024. K2: A foundation language model for geoscience knowledge understanding and utilization. In: Proceedings of the 17th ACM International Conference on Web Search and Data Mining. pp. 161–170.
Franch-Pardo, I., et al., 2020. Spatial analysis and gis in the study of covid-19. A review. Sci. Total Environ. 739, 140033.
Gao, S., 2020. A review of recent researches and reflections on geospatial artificial intelligence. Geomat. Inf. Sci. Wuhan Univ. 45 (12), 1865–1874.

Gao, S., Goodchild, M.F., 2013. Asking spatial questions to identify gis functionality. In: 2013 Fourth International Conference on Computing for Geospatial Research and Application. IEEE, pp. 106–110.
Ge, Y., et al., 2024. Openagi: When llm meets domain experts. Adv. Neural Inf. Process. Syst. 36.
Goodchild, M.F., 2000. Gis and transportation: status and challenges. GeoInformatica 4, 127–139.
Goodchild, M.F., 2016. Gis in the era of big data. Cybergeo: Eur. J. Geogr..
Guo, T., et al., 2024. Large language model based multi-agents: A survey of progress and challenges. arXiv preprint arXiv:2402.01680.
Houlsby, N., et al., 2019. Parameter-efficient transfer learning for nlp. In: International Conference on Machine Learning. PMLR, pp. 2790–2799.
Hu, E.J., et al., 2021. Lora: Low-rank adaptation of large language models. arXiv preprint arXiv:2106.09685.
Hu, Y., et al., 2023. Geo-knowledge-guided gpt models improve the extraction of location descriptions from disaster-related social media messages. Int. J. Geogr. Inf. Sci. 37 (11), 2289–2318.
Jamhiri, B., Shadabfar, M., Jalal, F.E., 2023a. Spatial uncertainty quantification of desiccation cracks in clays with limit state-adjusted linear elasticity. Model. Earth Syst. Environ. 9 (2), 2285–2303.
Jamhiri, B., et al., 2023b. Probabilistic estimation of thermal crack propagation in clays with gaussian processes and random fields. Geomech. Energy Environ. 34, 100454.
Janowicz, K., et al., 2020. Geoai: spatially explicit artificial intelligence techniques for geographic knowledge discovery and beyond.
Jiang, W., et al., 2024a. Change detection of multisource remote sensing images: a review. Int. J. Digit. Earth 17 (1), 2398051.
Jiang, Y., et al., 2024b. Urbanllm: Autonomous urban activity planning and management with large language models. arXiv preprint arXiv:2406.12360.
Kaplan, J., et al., 2020. Scaling laws for neural language models. arXiv preprint arXiv:2001.08361.
Lester, B., Al-Rfou, R., Constant, N., 2021. The power of scale for parameter-efficient prompt tuning. arXiv preprint arXiv:2104.08691.
Li, X.L., Liang, P., 2021. Prefix-tuning: Optimizing continuous prompts for generation. arXiv preprint arXiv:2101.00190.
Li, Z., Ning, H., 2023. Autonomous gis: the next-generation ai-powered gis. Int. J. Digit. Earth 16 (2), 4668–4686.
Liu, Q., et al., 2024. When moe meets llms: Parameter efficient fine-tuning for multi-task medical applications. In: Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval. pp. 1104–1114.
Magesh, N., Chandrasekar, N., Kaliraj, S., 2012. A gis based automated extraction tool for the analysis of basin morphometry. Bonfring Int. J. Ind. Eng. Manag. Sci. 2 (1), 32–35.
Maguire, D.J., 1991. An overview and definition of gis. Geogr. Inf. Syst.: Princ. Appl. 1 (1), 9–20.
Mai, G., et al., 2024. On the opportunities and challenges of foundation models for geoai (vision paper). ACM Trans. Spatial Algorithms Syst..
Mericskay, B., 2018. Automation of workflows for the installation of a wind farm. QGIS Appl. Territ. Plan. 3, 125–168.
Ouyang, L., et al., 2022. Training language models to follow instructions with human feedback. Adv. Neural Inf. Process. Syst. 35, 27730–27744.
Park, S., et al., 2019. A qgis-based graphical user interface for application and evaluation of swat-modflow models. Environ. Model. Softw. 111, 493–497.
Peng, J.L., et al., 2024. A survey of useful llm evaluation. arXiv preprint arXiv:2406.00936.
Polino, A., Pascanu, R., Alistarh, D., 2018. Model compression via distillation and quantization. arXiv preprint arXiv:1802.05668.
Qin, Y., et al., 2023. Toolllm: Facilitating large language models to master 16000+ real-world apis. Available from: https://arxiv.org/abs/2307.16789.
Scheider, S., et al., 2021. Geo-analytical question-answering with gis. Int. J. Digit. Earth 14 (1), 1–14.
Schick, T., et al., 2024. Toolformer: Language models can teach themselves to use tools. Adv. Neural Inf. Process. Syst. 36.
Taori, R., et al., 2023. Alpaca: A Strong, Replicable Instruction-Following Model. Vol. 3, Stanford Center for Research on Foundation Models, p. 7, https://crfm.stanford.edu/2023/03/13/alpaca.html. (6).
Thirunavukarasu, A.J., et al., 2023. Large language models in medicine. Nat. Med. 29 (8), 1930–1940.
Touvron, H., et al., 2023. Llama 2: Open foundation and fine-tuned chat models. Available from: https://arxiv.org/abs/2307.09288.
Wang, Y., et al., 2022. Self-instruct: Aligning language models with self-generated instructions. arXiv preprint arXiv:2212.10560.
Wang, J., et al., 2024a. Empowering legal citation recommendation via efficient instruction-tuning of pre-trained language models. In: European Conference on Information Retrieval. Springer, pp. 310–324.
Wang, J., et al., 2024b. Leveraging visual language model and generative diffusion model for zero-shot sar target recognition. Remote Sens. 16 (16), 2927.
Wang, L., et al., 2024c. A survey on large language model based autonomous agents. Front. Comput. Sci. 18 (6), 186345.
Wei, Cheng, Zhang, Yifan, Zhao, Xinru, Zeng, Ziyi, Wang, Zhiyun, Lin, Jianfeng, Guan, Qingfeng, Yu, Wenhao, 2024. Geotool-gpt: a trainable method for facilitating large language models to master gis tools. Int. J. Geograph. Inform. Sci. 1–25.
Wright, D.J., Goodchild, M.F., Proctor, J.D., 1997. Gis: Tool or science? demystifying the persistent ambiguity of gis as tool versus science. Ann. Assoc. Am. Geogr. 346–362.
Wu, C., et al., 2023. Visual chatgpt: Talking, drawing and editing with visual foundation models. arXiv preprint arXiv:2303.04671.
Xu, D., et al., 2024a. Editing factual knowledge and explanatory ability of medical large language models. In: Proceedings of the 33rd ACM International Conference on Information and Knowledge Management. pp. 2660–2670.
Xu, W., et al., 2024b. Rs-agent: Automating remote sensing tasks through intelligent agents. arXiv preprint arXiv:2406.07089.
Yang, R., et al., 2024. Gpt4tools: Teaching large language model to use tools via self-instruction. Adv. Neural Inf. Process. Syst. 36.
Yao, S., et al., 2022. React: Synergizing reasoning and acting in language models. arXiv preprint arXiv:2210.03629.
Yeh, A.G., 1999. Urban planning and gis. Geogr. Inf. Syst. 2 (877–888), 1.
Yuan, L., Yu, Z., Luo, W., 2019. Towards the next-generation gis: A geometric algebra approach. Ann. GIS 25 (3), 195–206.
Yue, P., Jiang, L., 2014. Biggis: How big data can shape next-generation gis. In: 2014 the Third International Conference on Agro-Geoinformatics. IEEE, pp. 1–6.
Zerger, A., Smith, D.I., 2003. Impediments to using gis for real-time disaster decision support. Comput. Environ. Urban Syst. 27 (2), 123–141.
Zhang, Y., Yu, W., Zhu, D., 2022. Terrain feature-aware deep learning network for digital elevation model superresolution. ISPRS J. Photogramm. Remote Sens. 189, 143–162.
Zhang, S., et al., 2024a. Instruction tuning for large language models: A survey. Available from: https://arxiv.org/abs/2308.10792.
Zhang, W., et al., 2024b. A progressive framework of vision-language knowledge distillation and alignment for multilingual scene. arXiv preprint arXiv:2404.11249.
Zhang, Y., et al., 2024c. Mapgpt: an autonomous framework for mapping by integrating large language model and cartographic tools. Cartogr. Geogr. Inf. Sci. 1–27.
Zhang, Y., et al., 2024d. Bb-geogpt: A framework for learning a large language model for geographic information science. Inf. Process. Manage. 61 (5), 103808.
Zhang, Y., et al., 2024e. Geogpt: An assistant for understanding and processing geospatial tasks. Int. J. Appl. Earth Obs. Geoinf. 131, 103976.
Zhao, T., et al., 2024. Artificial intelligence for geoscience: Progress, challenges and perspectives. Innovation.
Zhu, A.X., et al., 2021. Next generation of gis: must be easy. Ann. GIS 27 (1), 71–86.
Zhuang, Y., et al., 2024. Toolqa: A dataset for llm question answering with external tools. Adv. Neural Inf. Process. Syst. 36.