Spatial-Agent: Agentic Geo-spatial Reasoning with Scientific Core Concepts

Riyang Bao[1*], Cheng Yang[2*], Dazhou Yu[1], Zhexiang Tang[2], Gengchen Mai[3], Liang Zhao[1†]

[1]Emory University, [2]Rutgers University, [3]University of Texas at Austin

{riyang.bao, dazhou.yu, liang.zhao}@emory.edu

{cheng.yang, zhexiang.tang}@rutgers.edu

gengchen.mai@austin.utexas.edu

Abstract

Geospatial reasoning is essential for real-world applications such as urban analytics, transportation planning, and disaster response. However, existing LLM-based agents often fail at genuine geospatial computation, relying instead on web search or pattern matching while hallucinating spatial relationships. We present **Spatial-Agent**, an AI agent grounded in foundational theories of spatial information science. Our approach formalizes geo-analytical question answering as a *concept transformation* problem, where natural-language questions are parsed into executable workflows represented as *GeoFlow Graphs*—directed acyclic graphs with nodes corresponding to spatial concepts and edges representing transformations. Drawing on spatial information theory, Spatial-Agent extracts spatial concepts, assigns functional roles with principled ordering constraints, and composes transformation sequences through template-based generation. Extensive experiments on MapEval-API and MapQA benchmarks demonstrate that Spatial-Agent significantly outperforms existing baselines including ReAct and Reflexion, while producing interpretable and executable geospatial workflows.

1 Introduction

Geospatial reasoning is essential for numerous real-world applications, including urban analytics, transportation planning, environmental monitoring, disaster response, and public health (Li et al., 2021, 2023; Mai et al., 2021, 2025). As geospatial datasets and modern GIS platforms continue to proliferate, users increasingly expect natural-language interfaces capable of handling complex geo-analytical questions (Yu et al., 2025; Scheider et al., 2020, 2021). However, despite the impressive capabilities of contemporary large language models (LLMs), we observe that current agent-style systems, including function-calling LLMs and commercial AI agents, do not perform genuine geospatial reasoning.

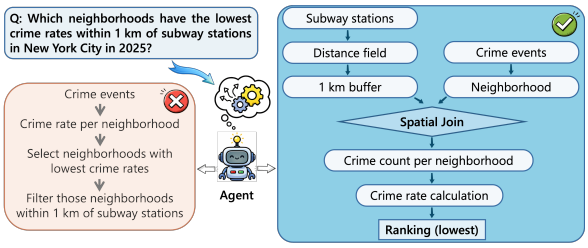

Figure 1: LLM-intuitive but incorrect workflow (left) vs. correct concept transformation (right). The incorrect workflow applies spatial constraints after aggregation; the correct workflow computes crime rates within the spatial context first.

Prior work shows that LLM-based agents lack inherent spatial awareness, relying on web search or textual pattern matching rather than computational spatial analysis (Yan et al., 2023; Zhang et al., 2025a). As illustrated in Figure 1, they hallucinate spatial relationships, fail on geometric or topological predicates, and cannot construct valid workflows (Wang et al., 2025; Ji et al., 2025).

This reflects a deeper limitation: geo-analytical questions require procedural, multi-step reasoning over spatial data, which fundamentally differs from declarative QA. Classical GIScience research has long emphasized that geographical phenomena are inherently computational rather than purely linguistic (Goodchild, 1992; Miller and Goodchild, 2015; Goodchild et al., 2007). Answering a geo-analytical question often involves: (i) identifying core spatial entities (Object, Event, Field, Network), (ii) selecting appropriate spatial operators (buffering, overlay, routing, aggregation), (iii) ordering operators into an executable workflow, and (iv) grounding abstract instructions into concrete GIS tools (e.g., PostGIS, ArcGIS, QGIS). These

[*]Equal contribution.

[†]Corresponding author.

Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 14896–14911

operations are geometric, topological, and sometimes spatiotemporal in nature, and thus beyond the representational power of language-only models.

A promising direction in GIScience suggests that geo-analytical questions encode implicit procedural knowledge. Kuhn's theory of *core concepts* (Kuhn, 2012) and subsequent work by Scheider et al. (2020) identify foundational building blocks of spatial information, such as *Object*, *Field*, *Event* and *Network*. Complementing these, GeoAnQu research (Xu et al., 2023) highlights *functional roles* (*Measure*, *Condition*, *Subcondition*, *Support*, *Extent*, etc.) that encode the procedural structure of geo-analytical questions. By combining core concepts and functional roles, GeoAnQu demonstrates that natural-language questions can be mapped into concept transformations, forming the basis of GIS workflows represented as directed acyclic graphs (DAGs). Yet this framework remains rule-based, offline, and detached from modern AI agent architectures.

In this paper, we ask: **What would it take for an AI agent to truly understand and execute geo-analytical questions?** Despite advances in tool-augmented LLMs, current agents remain limited in the geospatial domain. They often misinterpret spatial entities and relations, lack a procedural understanding of how geo-analytical tasks unfold, and cannot reliably map natural-language questions to coherent sequences of spatial operations. These systems also struggle to manage intermediate spatial states or ground their reasoning in computational GIS tools, which leads to answers that are largely descriptive rather than operational. As a result, existing agents fail to produce verifiable and executable geospatial analyses.

To address these challenges, we present **Spatial-Agent**, a geospatial AI agent grounded in foundational theories of spatial information. We formalize geo-analytical question answering as a *concept transformation* problem, where natural-language questions are parsed into executable workflows represented as *GeoFlow Graphs*. Drawing on core concepts and functional roles, Spatial-Agent establishes a principled intermediate representation that bridges language and computation. Specifically, the agent (i) extracts spatial concepts from questions and instantiates them as graph nodes, (ii) identifies functional roles that impose ordering constraints, (iii) composes transformation edges through a template-based approach that leverages recurring geo-analytical patterns, and (iv) executes the workflow via tool invocations, grounding its final response in verifiable computational results rather than parametric knowledge alone.

Our contributions are summarized as follows:

- We present Spatial-Agent, which enables geospatial reasoning by uncovering the implicit structure of spatial questions and generating coherent, executable workflows.
- We propose a compositional GeoFlow Graph generation approach based on macro-templates, capturing recurring geo-analytical patterns and improving structural validity via template matching and IO-port composition.
- Extensive evaluations show that Spatial-Agent delivers significantly better correctness, interpretability, and executable workflow generation than existing agent baselines, bridging the gap between natural-language reasoning and computational GIS.

Code and data are available at `https://github.com/ecerybao/Spatial-Agent`.

2 Related Work

Geospatial Question Answering. Geospatial question answering (Mai et al., 2021) is a subdomain of question answering focusing on questions that involve geographic entities, concepts, and/or require geospatial computation. Early geospatial QA systems focused on factoid-style questions over knowledge graphs (Mai et al., 2020). GeoQA (Punjani et al., 2018) and its successor GeoQA2 (Kefalidis et al., 2024) answer questions over DBpedia and YAGO2geo using template-based SPARQL query generation. The GeoQuestions1089 benchmark (Kefalidis et al., 2023) provides 1,089 questions with GeoSPARQL queries for evaluation. TourismQA (Contractor et al., 2021b,a) was constructed as a tourism-oriented geospatial QA dataset focusing on retrieving points of interest (POIs) from spatial databases based on text-to-SQL approaches according to specified geospatial and semantic constraints. More recently, MapQA (Li et al., 2025) was introduced as a similar text-to-SQL style POI retrieval benchmark with diverse geospatial constraints. However, these systems primarily handle declarative queries rather than procedural, multi-step geo-analytical reasoning. Scheider et al. (2021) and its follow-up work,

GeoAnQu (Xu et al., 2023), move toward geo-analytical questions by identifying functional roles and spatial concept transformations, but remain rule-based and offline.

LLM-based Agents and Tool Use. Recent advances in LLM agents have demonstrated impressive capabilities in tool-augmented reasoning. ReAct (Yao et al., 2022) introduced the thought-action-observation loop for interleaved reasoning and acting. Toolformer (Schick et al., 2023) enables self-supervised tool use learning, while CodeAct (Wang et al., 2024) shows that generating executable code outperforms JSON-based actions by up to 20%. In the geospatial domain, LLM-Geo (Li and Ning, 2023) and GeoGPT (Zhang et al., 2024) demonstrate autonomous GIS capabilities using GPT-4 for geoprocessing workflow generation and code execution. GeoAgent (Chen et al., 2024) integrates RAG with Monte Carlo Tree Search for geospatial data processing, and GTChain (Zhang et al., 2025b) fine-tunes LLaMA for geospatial tool-use chains.

Spatial Core Concepts and Workflow Composition. Kuhn's theory of core concepts (Kuhn, 2012) provides foundational building blocks for spatial information, including *Object*, *Field*, *Event*, and *Network*. Scheider et al. (2020) formalized these into the Core Concept Data Type (CCD) ontology, enabling semantic constraints for GIS workflow automation. Kruiger et al. (2021) demonstrated that loose programming with CCD types can automatically construct valid workflows for tasks like accessibility assessment and spatial interpolation. In parallel, neural program synthesis (Devlin et al., 2017; Zhong et al., 2023) has shown success in generating structured programs from specifications.

3 Spatial-Agent

3.1 Problem formulation of Agentic Geo-Spatial Reasoning

Unlike existing LLM-based agents that rely on general-purpose planning and reasoning, Spatial-Agent is designed around a key insight: geo-analytical questions require grounding natural-language semantics into computational spatial representations. This poses three fundamental challenges that distinguish geo-analytical reasoning from general-purpose tasks:

- **Semantic Domain vs. Spatial Domain.** LLM agents represent spatial relationships as linguistic patterns rather than geometric structures. For instance, "airport" and "terminal" are linguistically interchangeable, yet spatially the terminal is contained within the airport—a distinction critical for distance computation but invisible to language models.
- **Cognitive Reasoning vs. Spatial Reasoning.** LLMs treat geospatial tools as black-box APIs without understanding the procedural structure of geo-analytical questions. They lack a principled way to decompose spatial queries into ordered operations, often producing invalid workflows.
- **Spatial Orchestration vs. Spatial Execution.** Even when LLMs understand individual spatial operations, they struggle to orchestrate them into executable workflows where intermediate states must be tracked and data dependencies respected.

To address these challenges, Spatial-Agent introduces: (1) explicit spatial grounding that transforms semantic descriptions into computational representations with precise geometric meanings; (2) a theory-driven framework leveraging GIScience core concepts and functional roles to decompose geo-analytical questions; and (3) the GeoFlow Graph as an intermediate representation bridging spatial knowledge with agent execution.

System Overview. As illustrated in Figure 2, Spatial-Agent operates through a multi-stage pipeline: (1) *Spatial Information Theory Analysis* (§3.2) extracts core concepts from the question and groups them by functional roles; (2) *Concept Transformation Drafting* retrieves templates to define transformation patterns between concepts; (3) *GeoFlow Graph Construction* (§3.3) assembles transformations into an ordered graph following role-based precedence constraints; (4) *GeoFlow Graph Factorization & Tool Mapping* converts the graph into an executable form with concrete operators; and (5) the agent executes the workflow and generates a grounded response.

3.2 Geospatial Concept Grounding

To ground semantic concepts in spatial analysis, we must map abstract, non-spatial concepts from their original domains to concrete geo-objects in

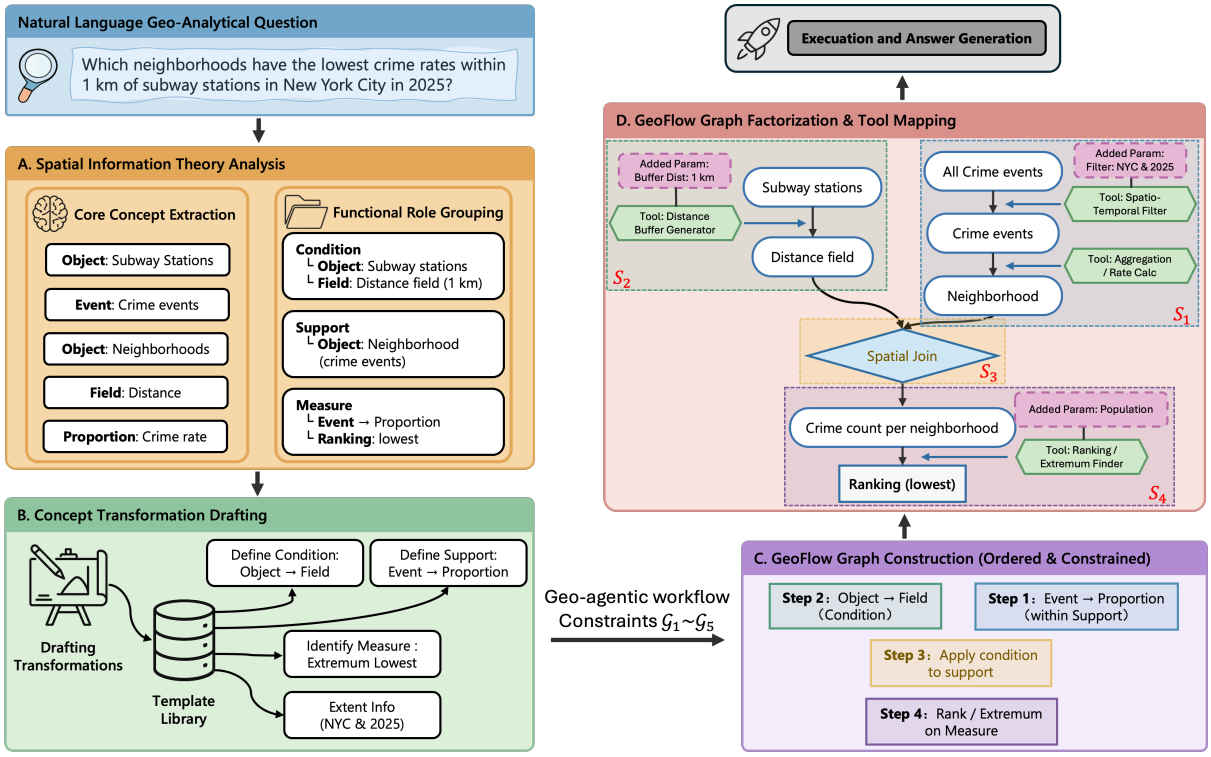

Figure 2: Overview of Spatial-Agent: (A) Spatial information theory analysis extracts core concepts and assigns functional roles; (B) Concept transformation drafting composes templates from the library; (C) GeoFlow Graph construction produces an ordered and constrained graph; (D) Graph factorization maps to executable tools for execution and answer generation.

the spatial domain, and subsequently derive the corresponding geographic relations among them. This grounding process requires two complementary formalisms: *core concepts* that define what spatial entities exist, and *functional roles* that specify how they participate in analysis. GIScience has long recognized that such grounding follows principled patterns governed by geospatial semantics (Goodchild, 1992; Kuhn, 2012; Mai et al., 2021; Scheider et al., 2021).

To formalize this grounding process, we represent the result as a *GeoFlow Graph*. Given a natural-language geo-analytical question q, our goal is to derive a GeoFlow Graph G that represents an executable geospatial workflow:

$$f : q \mapsto G \tag{1}$$

where $G = (V, E, \lambda, \rho)$ consists of concept nodes V, transformation edges $E \subseteq V \times V$, a concept labeling function $\lambda : V \to \mathcal{C}$ mapping nodes to spatial concepts (defined below based on spatial core concept theory), and a role assignment function $\rho : V \to \mathcal{R}$ assigning functional roles (defined below based on geo-analytical reasoning principles). Each directed edge $(v_i, v_j) \in E$ represents a *transformation*—a semantic change from one spatial concept to another (e.g., from a place name to coordinates via geocoding). Each transformation is realized by an *operator* $\omega \in \Omega$, a concrete computational function that takes input concepts and produces output concepts.

Here, we define the spatial concept space $\mathcal{C}$ as a set of *core concepts* that ground semantic representations into geographic primitives (Kuhn, 2012):

$$\mathcal{C} = \left\{ \begin{array}{l} \textsc{Location}, \textsc{Object}, \textsc{Field}, \textsc{Event}, \\ \textsc{Network}, \textsc{Amount}, \textsc{Proportion} \end{array} \right\} \tag{2}$$

These seven primitives, widely recognized in geographic information theory, capture the fundamental building blocks of geographic phenomena and provide a vocabulary for representing spatial entities extracted from natural-language questions (see Appendix A for detailed definitions and examples).

Crucially, we operationalize these abstract concepts into a mathematical framework that LLMs can manipulate. Each concept $c \in \mathcal{C}$ is not merely a label but carries associated type signatures, valid transformations, and composability rules. This formalization transforms the descriptive spatial theory into a computational substrate: the agent can query

concept compatibility, validate transformation sequences, and reason about workflow correctness through symbolic operations on $\mathcal{C}$.

While core concepts define *what* spatial entities are, we also need to specify *how* they participate in geo-analytical reasoning. To this end, we introduce *functional roles* to explicitly encode the procedural structure of geo-analytical questions (Xu et al., 2023):

$$\mathcal{R} = \left\{ \begin{matrix} \textsc{Extent}, \textsc{TExtent}, \textsc{SubCond}, \\ \textsc{Cond}, \textsc{Support}, \textsc{Measure} \end{matrix} \right\}. \quad (3)$$

We also distinguish between *contextual roles* (EXTENT, TEXTENT) and *procedural roles* (SUBCOND, COND, SUPPORT, MEASURE). Contextual roles constrain data collection but do not participate in the procedural ordering of transformations (e.g., "New York City" as EXTENT and "2025" as TEXTENT in Figure 2 (B)). For procedural roles, we define a precedence relation $\prec$ derived from the execution semantics of geo-analytical workflows: SUBCOND $\prec$ COND $\prec$ SUPPORT $\prec$ MEASURE.

This ordering reflects the inherent structure of geo-analytical workflows: sub-conditions restrict candidate entities, conditions constrain supports, supports establish the spatial basis, and measurements are the final outputs. By making this procedural structure explicit, the agent follows transformation orders derived from geographic first principles rather than hallucinating arbitrary operation sequences (see Appendix B for detailed role definitions).

3.3 Spatial Orchestration

With the concept space $\mathcal{C}$ and functional roles $\mathcal{R}$ established, we describe how to construct a GeoFlow Graph G from a question. The challenge is determining which transformation edges to include: an edge $(v_i, v_j) \in E$ should be added when the resolution of v_j depends on the output of v_i.

V includes both *explicit* concepts directly mentioned in the question and *implicit* concepts inferred to complete the workflow. For example, given "What is the driving time from my hotel to the nearest coffee shop?", the explicit concepts are HOTEL, COFFEESHOP, and DRIVINGTIME. However, computing driving time requires a ROADNETWORK (prerequisite data source) and an intermediate ROUTE connecting the two locations—neither of which is explicitly mentioned. The generation process (§3.4) is responsible for identifying and instantiating these implicit nodes.

GeoFlow Graph Factorization. GeoFlow Graphs capture concept-level transformations, where nodes represent spatial core concepts and edges denote semantic dependencies. However, these edges do not directly correspond to executable operators in an agentic system: many geo-analytical transformations jointly consume multiple input concepts, while the original graph encodes them as separate edges. To make GeoFlow Graphs executable, we reinterpret them as a factorized operator–concept hypergraph $G' = (V', E')$ such that there is a bijective mapping between them $G \leftrightarrow G'$, where V' are the nodes consisting of spatial core concept nodes and factor nodes. The concept nodes (e.g., the round nodes such as "Subway stations" and "Crime events" in Figure 2 (D)) are directly extracted from the question, while factor nodes (e.g., the pink boxes such as "Buffer Dist: 1 km" and "Population") represent supplementary parameters required for operator execution. Each operator takes concept nodes along with supplementary parameters as inputs and produces one or more downstream concept nodes as outputs. This factorization enables many-to-many relationships that traditional sequential graphs cannot directly express. The operators span geocoding, spatial search, routing, geometric computation, and trip optimization (see Appendix C for details).

Geo-Agentic Workflow Constraints. Based on fundamental GIScience principles (Tobler, 1970; Janowicz et al., 2012; Janowicz, 2012; Janowicz et al., 2022), we formalize the structural regularities of geographic phenomena as explicit constraints. A valid GeoFlow Graph must satisfy:

(1) **Acyclicity** ($\mathcal{G}_1$): $\mathcal{G}_1 = \{G \mid \nexists \text{ cycle in } E\}$, ensuring a valid topological execution order.

(2) **Role Ordering** ($\mathcal{G}_2$): $\mathcal{G}_2 = \{G \mid \forall (v_i, v_j) \in E : \rho(v_i) \preceq \rho(v_j)\}$.

(3) **Type Compatibility** ($\mathcal{G}_3$): $\mathcal{G}_3 = \{G \mid \forall (v_i, v_j) \in E : \tau_{\text{out}}(v_i) \subseteq \tau_{\text{in}}(v_j)\}$, where τ_{out} and τ_{in} denote output and input concept types.

(4) **Data Availability** ($\mathcal{G}_4$): $\mathcal{G}_4 = \{G \mid G \leftrightarrow G', \forall e \in E' : e \text{ is executable}\}$.

(5) **Connectivity** ($\mathcal{G}_5$): $\mathcal{G}_5 = \{G \mid \forall v \in V : \exists v_0 \in V_{\text{ext}}, v_m \in V_{\text{meas}} : \text{path}(v_0, v) \wedge \text{path}(v, v_m)\}$, where $V_{\text{ext}} = \{v : \rho(v) \in \{\textsc{Extent}, \textsc{TExtent}\}\}$ and $V_{\text{meas}} = \{v : \rho(v) = \textsc{Measure}\}$.

A GeoFlow Graph is *well-formed* if it satisfies all five constraints above, i.e., $G \in \bigcap_{i=1}^{5} \mathcal{G}_i$. We

formalize DAG assembly as a constraint satisfaction problem where multiple valid configurations may exist.

3.4 Retrieval-augmented Orchestration

While the agent can assemble GeoFlow Graphs from primitive operators subject to the constraints defined above, we observe that geo-analytical questions exhibit recurring structural patterns. To accelerate generation and improve accuracy, we adopt a compositional approach that leverages a library of pre-validated graph templates, each of which inherently satisfies the well-formedness constraints.

Template Library. We define a set of macro-templates $\mathcal{T} = \{g_1, \ldots, g_K\}$, where each template $g_k = (V_k, E_k, in_k, out_k)$ specifies a subgraph $g_k \subseteq G$ with designated input and output ports. As illustrated in Figure 2 (B), the Concept Transformation Drafting stage retrieves relevant templates from the library to define transformations such as OBJECT $\rightarrow$ FIELD and EVENT $\rightarrow$ PROPORTION, which are then composed into an ordered GeoFlow Graph (see Appendix E for the complete template library).

Given a question q, the LLM additionally generates the GeoFlow Graph guided by retrieved examples $\mathcal{E}_q$ from a question-graph store.

3.5 Learning with Geographic Constraints

While Spatial-Agent can operate with off-the-shelf LLMs via prompting, we optionally employ a two-stage fine-tuning strategy to further improve performance. In Stage 1, we apply Supervised Fine-Tuning (SFT) on question-concept pairs $\{(q_i, V_i)\}_{i=1}^{N}$, minimizing the negative log-likelihood $\mathcal{L}_{\text{SFT}} = -\sum_i \log p_\theta(V_i \mid q_i)$, where p_θ denotes the LLM parameterized by θ, to train the LLM to extract spatial concepts with their types and functional roles. In Stage 2, we use Direct Preference Optimization (DPO) to train the LLM to generate well-formed GeoFlow Graphs. For each question q, we construct preference pairs (G^+, G^-) where $G^+ \in \bigcap_{i=1}^{5} \mathcal{G}_i$ while $G^- \notin \bigcap_{i=1}^{5} \mathcal{G}_i$, and optimize:

$$\min_\theta \ \mathcal{L}_{\text{DPO}}, \quad \text{s.t. } G^+ \in \bigcap_{i=1}^{5} \mathcal{G}_i,\ G^- \notin \bigcap_{i=1}^{5} \mathcal{G}_i \quad (4)$$

where $\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(q,G^+,G^-)}\left[\log \sigma\left(\beta \cdot r_\theta\right)\right]$, $r_\theta = \log \frac{p_\theta(G^+|q)}{p_{\text{ref}}(G^+|q)} - \log \frac{p_\theta(G^-|q)}{p_{\text{ref}}(G^-|q)}$ is the reward margin, σ is the sigmoid function, β is the temperature, and p_{ref} is the reference model. This optional fine-tuning enables the model to better internalize geographic reasoning patterns (see Appendix D for details).

3.6 Execution and Response Generation

Given a well-formed GeoFlow Graph G, we first transform it to the factorized graph G' and then execute transformation steps in topological order. All intermediate states are recorded and provided to the agent for response generation, ensuring the output is grounded in computational results rather than hallucinated from learned priors (see Appendix F for the detailed algorithm).

4 Experiments

4.1 Experimental Setup

Datasets. We evaluate Spatial-Agent on two comprehensive geospatial reasoning benchmarks that collectively cover diverse task types, geographic regions, and reasoning capabilities. **MapEval-API** (Dihan et al., 2024) is the API-based evaluation from the MapEval benchmark, which requires agents to invoke map tools for geospatial reasoning. The dataset comprises four task categories: *Place Info* (retrieving place attributes), *Nearby* (finding nearby points of interest), *Routing* (computing directions and distances), and *Trip* (multi-stop travel planning). The benchmark spans 180 cities across 54 countries, providing diverse geographic coverage for evaluating tool-augmented spatial reasoning. **MapQA** (Li et al., 2025) is an open-domain geospatial QA dataset with 3,154 question-answer pairs spanning nine question types, constructed from OpenStreetMap data covering Southern California and Illinois.

Baselines. We compare Spatial-Agent against the following methods: (1) **Direct LLM**, which directly prompts the language model without agent scaffolding; (2) **ReAct** (Yao et al., 2022), an agent that interleaves reasoning and acting through thought-action-observation loops; (3) **Reflexion** (Shinn et al., 2023), an agent that learns from execution failures through self-reflection and memory; and (4) **Plan-and-Solve** (Wang et al., 2023), a prompting strategy that first generates an overall plan before execution.

4.2 Main Results

Results on MapEval-API. Table 1 presents the results on the MapEval-API benchmark across four

task categories. Spatial-Agent consistently outperforms all baselines across different backbone LLMs. For closed-source models, Spatial-Agent with GPT-4o-mini achieves an overall accuracy of 45.15%, representing a 96.30% relative improvement over the MapEval API baseline (23.00%). The improvement is particularly pronounced on *Place Info* (+149.91%) and *Nearby* (+133.26%) tasks, where structured tool invocation and spatial reasoning are critical. When equipped with GPT-5, Spatial-Agent reaches the best overall accuracy of 71.88%, with strong performance across all categories, especially on *Routing* (75.76%) and *Trip* (77.61%) tasks that require multi-step planning.

For open-source LLMs, Spatial-Agent demonstrates competitive performance. Qwen2.5-72B-Instruct achieves the best overall accuracy of 53.41% among open-source models, with the highest *Trip* accuracy (61.19%), while Qwen2.5-32B-Instruct achieves the best *Routing* accuracy (50.00%). With LLaMA-70B, our method achieves 47.77% overall accuracy (+26.82% over baseline), with substantial improvements on *Place Info* (+35.29%) and *Nearby* (+62.96%). We observe that *Routing* tasks remain challenging for LLaMA-70B (-17.85%), suggesting that complex navigation queries benefit more from models with stronger reasoning capabilities.

Results on MapQA. Table 2 presents the results on the MapQA benchmark across six question types. For closed-source models, Spatial-Agent (GPT-4o-mini) achieves the best overall accuracy of 61.45%, substantially outperforming Direct LLM (13.55%), ReAct (43.79%), and Reflexion (53.79%). The improvement is particularly notable on *Amenities-Around* and *Amenities-Around-Specific* tasks, demonstrating the effectiveness of our spatial reasoning framework for complex location-based queries.

For open-source models, Spatial-Agent with LLaMA-70B achieves the highest overall accuracy of 62.45%, with the best performance on *Amenities* (84.00%). Qwen2.5-72B-Instruct achieves comparable overall accuracy (61.45%) and the highest *Amenities-Around-Specific* accuracy (78.00%). Notably, the open-source Spatial-Agent variants achieve competitive or superior performance compared to the closed-source GPT-4o-mini configuration, demonstrating the generalizability of our framework across different model families.

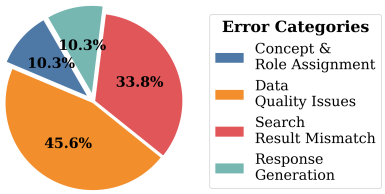

Figure 3: Distribution of error categories in Spatial-Agent. Data Quality Issues (45.6%) and Search Result Mismatch (33.8%) account for the majority of errors, both occurring during the execution stage.

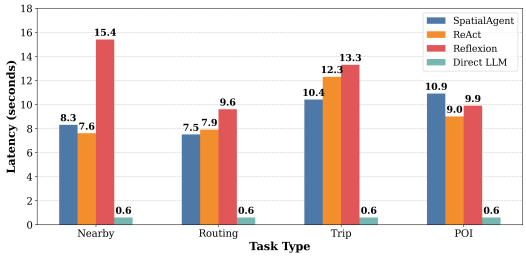

Figure 4: Average latency per query (seconds) across task types. All methods use GPT-4o-mini.

4.3 Analysis

Error Analysis. To understand the limitations of Spatial-Agent, we manually analyzed 68 incorrect predictions on MapEval-API, categorizing errors according to our four-stage pipeline (Figure 3).

We categorize errors into four types (Figure 3): Data Quality Issues (45.6%) and Search Result Mismatch (33.8%) occur during execution when external APIs return incomplete data or mismatched results; Concept & Role Assignment (10.3%) involves misidentifying concepts or functional roles; Response Generation (10.3%) occurs when correct execution leads to incorrect answer selection. Notably, no errors originated from GeoFlow Graph construction itself, validating our template-based approach. This confirms that the primary bottleneck lies in external API interactions rather than reasoning components.

Latency Analysis. Figure 4 compares response latency across methods using GPT-4o-mini. Direct LLM achieves the lowest latency (0.6s) but poor accuracy due to lack of grounding. Among agentic methods, Spatial-Agent shows competitive latency: fastest on Routing (7.5s), comparable to ReAct on Nearby (8.3s vs 7.6s) and Trip (10.4s vs 12.3s), slightly slower on POI (10.9s vs 9.0s). Reflexion consistently exhibits the highest latency due to its iterative self-reflection mechanism.

Method	Overall	Place Info	Nearby	Routing	Trip
Closed-Source LLMs					
Direct LLM (GPT-4o-mini)	32.23	45.31	32.53	22.73	28.36
ReAct (GPT-4o-mini)	32.98	60.94	22.89	22.73	25.37
Reflexion (GPT-4o-mini)	38.29	65.63	22.89	30.30	34.33
MapEval API (GPT-3.5-Turbo)	27.33	39.06	22.89	33.33	19.40
MapEval API (GPT-4o-mini)	23.00	28.13	14.46	13.64	43.28
Spatial-Agent (GPT-3.5-Turbo)	34.61 ↑26.64%	57.81 ↑48.03%	26.51 ↑15.81%	36.36 ↑9.09%	25.37 ↑30.77%
Spatial-Agent (GPT-4o-mini)	45.15 ↑96.30%	70.31 ↑149.91%	33.73 ↑133.26%	30.30 ↑122.14%	46.27 ↑6.91%
Spatial-Agent (GPT-5)	**71.88**	**85.94**	**53.01**	**75.76**	**77.61**
Open-Source LLMs					
MapEval API (LlaMA-70B)	37.67	53.13	32.53	42.42	31.34
Spatial-Agent (LlaMA-70B)	47.77 ↑26.82%	**71.88** ↑35.29%	**53.01** ↑62.96%	34.85 ↓17.85%	31.34 ↑0.00%
Spatial-Agent (Qwen2.5-72B-Instruct)	**53.41**	68.75	39.76	43.94	**61.19**
Spatial-Agent (Qwen2.5-32B-Instruct)	52.35	**71.88**	39.76	**50.00**	47.76
Spatial-Agent (Gemma-2-9B)	22.70	31.25	14.46	19.70	25.37

Table 1: Results on MapEval-API. Accuracy (%) across four task categories. **Bold**/underline: best/second-best. ↑/↓: improvement/decrease over baseline.

Type	Method	Overall	Adj	Amen	Amen-A	Amen-AS	Cmp-Cl	Dist
Closed-Source	Direct LLM (GPT-4o-mini)	13.55	6.00	36.00	4.00	0.00	35.29	0.00
	ReAct (GPT-4o-mini)	43.79	**56.00**	28.00	16.00	64.00	**64.71**	34.00
	Reflexion (GPT-4o-mini)	53.79	50.00	80.00	30.00	64.00	**64.71**	34.00
	Spatial-Agent (GPT-4o-mini)	**61.45**	52.00	**82.00**	**56.00**	**74.00**	**64.71**	**40.00**
Open-Source	Spatial-Agent (Qwen2.5-72B-Instruct)	61.45	48.00	76.00	64.00	78.00	64.71	38.00
	Spatial-Agent (LlaMA-70B)	62.45	54.00	84.00	58.00	76.00	64.71	38.00

Table 2: Results on the MapQA benchmark. Accuracy (%) across six question types. **Bold**/underline: best/second-best within each model group. Adj: Adjacent; Amen: Amenities; Amen-A: Amenities-Around; Amen-AS: Amenities-Around-Specific; Cmp-Cl: Compare-Closer; Dist: Distance.

Cost Analysis. We compare token consumption across methods: Spatial-Agent uses 9,185 input and 1,451 output tokens per query; Reflexion consumes the most (10,964 input, 1,271 output); ReAct uses fewer (7,354 input, 735 output); Direct LLM is minimal (1,500 input, 50 output). With GPT-4o-mini pricing ($0.15/1M input, $0.60/1M output), all methods cost below $0.003 per query, with Spatial-Agent at $0.0022 offering the best accuracy-cost trade-off.

4.4 Ablation Study

Effect of Fine-tuning. We evaluate our two-stage fine-tuning on Qwen-14B (Table 3, bottom). The base model achieves 49.59% accuracy. SFT alone improves to 56.84% (+14.6%) by learning concept extraction and role assignment; DPO alone yields 55.13% (+11.2%) through preference learning for graph construction. However, both single-stage strategies cause drops on *Route* tasks, suggesting overfitting on certain patterns. The combined SFT+DPO achieves the best accuracy of 60.58% (+22.2%) with consistent improvements across all categories, especially *Trip* (+50.0%), indicating that multi-step planning benefits from both accurate concept extraction and valid graph composition.

Effect of Template Composition. We compare Spatial-Agent with and without templates using GPT-4o-mini (Table 3, top). Removing templates drops accuracy from 45.15% to 39.32% (-12.9%), with consistent degradation across categories: *POI* (-17.8%), *Near* (-14.3%), *Route* (-10.0%), *Trip* (-6.5%). This validates that templates encoding recurring geo-analytical patterns improve validity.

5 Conclusion

We present Spatial-Agent, a geospatial AI agent that formalizes geo-analytical question answering as a concept transformation problem grounded in GIScience theory. The GeoFlow Graph representation encodes spatial concepts, functional roles, and well-formedness constraints, enabling structured reasoning over geographic workflows. Our template-based compositional generation leverages recurring geo-analytical patterns to improve structural validity, while SFT+DPO fine-tuning enables models to internalize geographic constraints. Experiments demonstrate that Spatial-Agent significantly outperforms existing agent baselines. Er-

Method	All	POI	Near	Route	Trip
Plan-and-Solve (4o-mini)	41.17	60.94	30.12	**31.82**	41.79
Spatial-Agent (4o-mini)	**45.15**	**70.31**	**33.73**	30.30	**46.27**
w/o Template	39.32 ↓12.9%	57.81 ↓17.8%	28.92 ↓14.3%	27.27 ↓10.0%	43.28 ↓6.5%
Spatial-Agent (Qwen-14B)	49.59	64.06	46.99	48.48	38.81
w/ SFT only	56.84 ↑14.6%	79.69 ↑24.4%	46.99 ↑0.0%	45.45 ↓6.3%	55.22 ↑42.3%
w/ DPO only	55.13 ↑11.2%	81.25 ↑26.8%	44.58 ↓5.1%	43.94 ↓9.4%	50.75 ↑30.8%
w/ SFT + DPO	**60.58** ↑22.2%	**84.38** ↑31.7%	**48.19** ↑2.6%	**51.52** ↑6.3%	**58.21** ↑50.0%

Table 3: Ablation study results on MapEval-API. Accuracy (%) across task categories. **Top**: effect of template composition with GPT-4o-mini backbone. **Bottom**: effect of two-stage fine-tuning (SFT, DPO) on Qwen-14B. **Bold**/underline: best/second-best within each block. ↑/↓: improvement/decrease relative to the in-block reference row. POI: Place Info; Near: Nearby; Route: Routing.

ror analysis reveals that the primary bottleneck lies in external API interactions rather than reasoning components, validating the effectiveness of our structured spatial reasoning approach.

Limitations

Several limitations remain. First, the framework's accuracy is bounded by the reliability of external geospatial APIs, with most errors occurring during execution due to data quality issues. Second, the template library may not cover all question types, requiring from-scratch generation for novel patterns. Third, the fine-tuning approach requires annotated data that demands significant effort to scale to new domains. Finally, our evaluation focuses on English-language urban environments; performance on specialized geographic domains remains unexplored.

Acknowledgments

This work was partially supported by the NSF Grant No. 2403312, No. 2414115, No. 2007716, No. 2007976, No. 1942594, No. 1907805, Cisco Faculty Research Award, Sony Research Award, and NIH Grant No. R01AG089806. Gengchen Mai is supported by the NSF under Grant No. 2521631.

References

Yuxing Chen, Weijie Wang, Sylvain Lobry, and Camille Kurtz. 2024. An llm agent for automatic geospatial data analysis. *arXiv preprint arXiv:2410.18792*.

Danish Contractor, Shashank Goel, Mausam, and Parag Singla. 2021a. Joint spatio-textual reasoning for answering tourism questions. In *Proceedings of the Web Conference 2021*, pages 1978–1989.

Danish Contractor, Krunal Shah, Aditi Partap, Parag Singla, and Mausam Mausam. 2021b. Answering poi-recommendation questions using tourism reviews. In *Proceedings of the 30th ACM International Conference on Information & Knowledge Management*, pages 281–291.

Jacob Devlin, Jonathan Uesato, Surya Bhupatiraju, Rishabh Singh, Abdel-rahman Mohamed, and Pushmeet Kohli. 2017. Robustfill: Neural program learning under noisy i/o. In *International conference on machine learning*, pages 990–998. PMLR.

Mahir Labib Dihan, Md Tanvir Hassan, Md Tanvir Parvez, Md Hasebul Hasan, Md Almash Alam, Muhammad Aamir Cheema, Mohammed Eunus Ali, and Md Rizwan Parvez. 2024. Mapeval: A map-based evaluation of geo-spatial reasoning in foundation models. *arXiv preprint arXiv:2501.00316*.

Michael F Goodchild. 1992. Geographical information science. *International journal of geographical information systems*, 6(1):31–45.

Michael F Goodchild, May Yuan, and Thomas J Cova. 2007. Towards a general theory of geographic representation in gis. *International journal of geographical information science*, 21(3):239–260.

Krzysztof Janowicz. 2012. Observation-driven geo-ontology engineering. *Transactions in GIS*, 16(3):351–374.

Krzysztof Janowicz, Pascal Hitzler, Wenwen Li, Dean Rehberger, Mark Schildhauer, Rui Zhu, Cogan Shimizu, Colby Fisher, Ling Cai, Gengchen Mai, and 1 others. 2022. Know, know where, knowwheregraph: A densely connected, cross-domain knowledge graph and geo-enrichment service stack for applications in environmental intelligence. *AI Magazine*, 43(1):30–39.

Krzysztof Janowicz, Simon Scheider, Todd Pehle, and Glen Hart. 2012. Geospatial semantics and linked spatiotemporal data–past, present, and future. *Semantic Web*, 3(4):321–332.

Yuhan Ji, Song Gao, Ying Nie, Ivan Majić, and Krzysztof Janowicz. 2025. Foundation models for geospatial reasoning: assessing the capabilities of large language models in understanding geometries and topological spatial relations. *Interna-*

tional Journal of Geographical Information Science, 39(9):1866–1903.

Sergios-Anestis Kefalidis, Dharmen Punjani, Eleni Tsalapati, Konstantinos Plas, Maria-Aggeliki Pollali, Pierre Maret, and Manolis Koubarakis. 2024. The question answering system geoqa2 and a new benchmark for its evaluation. *International Journal of Applied Earth Observation and Geoinformation*, 134:104203.

Sergios-Anestis Kefalidis, Dharmen Punjani, Eleni Tsalapati, Konstantinos Plas, Mariangela Pollali, Michail Mitsios, Myrto Tsokanaridou, Manolis Koubarakis, and Pierre Maret. 2023. Benchmarking geospatial question answering engines using the dataset geoquestions1089. In *International semantic web conference*, pages 266–284. Springer.

Johannes F Kruiger, Vedran Kasalica, Rogier Meerlo, Anna-Lena Lamprecht, Enkhbold Nyamsuren, and Simon Scheider. 2021. Loose programming of gis workflows with geo-analytical concepts. *Transactions in GIS*, 25(1):424–449.

Werner Kuhn. 2012. Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, 26(12):2267–2276.

Haonan Li, Ehsan Hamzei, Ivan Majic, Hua Hua, Jochen Renz, Martin Tomko, Maria Vasardani, Stephan Winter, and Timothy Baldwin. 2021. Neural factoid geospatial question answering. *Journal of Spatial Information Science*, (23):65–90.

Haonan Li, Martin Tomko, and Timothy Baldwin. 2023. Location aware modular biencoder for tourism question answering. In *Findings of the Association for Computational Linguistics: IJCNLP-AACL 2023 (Findings)*, pages 95–109.

Zekun Li, Malcolm Grossman, Mihir Kulkarni, Muhao Chen, Yao-Yi Chiang, and 1 others. 2025. Mapqa: Open-domain geospatial question answering on map data. *arXiv preprint arXiv:2503.07871*.

Zhenlong Li and Huan Ning. 2023. Autonomous gis: the next-generation ai-powered gis. *International Journal of Digital Earth*, 16(2):4668–4686.

Gengchen Mai, Krzysztof Janowicz, Ling Cai, Rui Zhu, Blake Regalia, Bo Yan, Meilin Shi, and Ni Lao. 2020. Se-kge: A location-aware knowledge graph embedding model for geographic question answering and spatial semantic lifting. *Transactions in GIS*, 24(3):623–655.

Gengchen Mai, Krzysztof Janowicz, Rui Zhu, Ling Cai, and Ni Lao. 2021. Geographic question answering: challenges, uniqueness, classification, and future directions. *AGILE: GIScience series*, 2:8.

Gengchen Mai, Yiqun Xie, Xiaowei Jia, Ni Lao, Jinmeng Rao, Qing Zhu, Zeping Liu, Yao-Yi Chiang, and Junfeng Jiao. 2025. Towards the next generation of geospatial artificial intelligence. *International Journal of Applied Earth Observation and Geoinformation*, 136:104368.

Harvey J Miller and Michael F Goodchild. 2015. Data-driven geography. *GeoJournal*, 80(4):449–461.

Dharmen Punjani, Kuldeep Singh, Andreas Both, Manolis Koubarakis, Iosif Angelidis, Konstantina Bereta, Themis Beris, Dimitris Bilidas, Theofilos Ioannidis, Nikolaos Karalis, and 1 others. 2018. Template-based question answering over linked geospatial data. In *Proceedings of the 12th workshop on geographic information retrieval*, pages 1–10.

Simon Scheider, Rogier Meerlo, Vedran Kasalica, and Anna-Lena Lamprecht. 2020. Ontology of core concept data types for answering geo-analytical questions. *Journal of Spatial Information Science*, (20):167–201.

Simon Scheider, Enkhbold Nyamsuren, Han Kruiger, and Haiqi Xu. 2021. Geo-analytical question-answering with gis. *International Journal of Digital Earth*, 14(1):1–14.

Timo Schick, Jane Dwivedi-Yu, Roberto Dessì, Roberta Raileanu, Maria Lomeli, Eric Hambro, Luke Zettlemoyer, Nicola Cancedda, and Thomas Scialom. 2023. Toolformer: Language models can teach themselves to use tools. *Advances in Neural Information Processing Systems*, 36:68539–68551.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language agents with verbal reinforcement learning. *Advances in neural information processing systems*, 36:8634–8652.

Waldo R Tobler. 1970. A computer movie simulating urban growth in the detroit region. *Economic geography*, 46(sup1):234–240.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: long papers)*, pages 2609–2634.

Shengyuan Wang, Jie Feng, Tianhui Liu, Dan Pei, and Yong Li. 2025. Mitigating geospatial knowledge hallucination in large language models: Benchmarking and dynamic factuality aligning. In *Findings of the Association for Computational Linguistics: EMNLP 2025*, pages 870–888.

Xingyao Wang, Yangyi Chen, Lifan Yuan, Yizhe Zhang, Yunzhu Li, Hao Peng, and Heng Ji. 2024. Executable code actions elicit better llm agents. In *Forty-first International Conference on Machine Learning*.

Haiqi Xu, Enkhbold Nyamsuren, Simon Scheider, and Eric Top. 2023. A grammar for interpreting geo-analytical questions as concept transformations. *International Journal of Geographical Information Science*, 37(2):276–306.

He Yan, Xinyao Hu, Xiangpeng Wan, Chengyu Huang, Kai Zou, and Shiqi Xu. 2023. Inherent limitations of llms regarding spatial information. *arXiv preprint arXiv:2312.03042*.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik R Narasimhan, and Yuan Cao. 2022. React: Synergizing reasoning and acting in language models. In *The eleventh international conference on learning representations*.

Dazhou Yu, Riyang Bao, Gengchen Mai, and Liang Zhao. 2025. Spatial-rag: Spatial retrieval augmented generation for real-world spatial reasoning questions. *arXiv e-prints*, pages arXiv–2502.

Qianheng Zhang, Song Gao, Chen Wei, Yibo Zhao, Ying Nie, Ziru Chen, Shijie Chen, Yu Su, and Huan Sun. 2025a. Geoanalystbench: A geoai benchmark for assessing large language models for spatial analysis workflow and code generation. *Transactions in GIS*, 29(7):e70135.

Yifan Zhang, Jingxuan Li, Zhiyun Wang, Zhengting He, Qingfeng Guan, Jianfeng Lin, and Wenhao Yu. 2025b. Geospatial large language model trained with a simulated environment for generating tool-use chains autonomously. *International Journal of Applied Earth Observation and Geoinformation*, 136:104312.

Yifan Zhang, Cheng Wei, Zhengting He, and Wenhao Yu. 2024. Geogpt: An assistant for understanding and processing geospatial tasks. *International Journal of Applied Earth Observation and Geoinformation*, 131:103976.

Linghan Zhong, Ryan Lindeborg, Jesse Zhang, Joseph J Lim, and Shao-Hua Sun. 2023. Hierarchical neural program synthesis. *arXiv preprint arXiv:2303.06018*.

A Core Spatial Concepts

Following Kuhn's theory of core concepts in spatial information (Kuhn, 2012), we define a fixed set of primitive spatial concepts that constitute the semantic foundation of geo-analytical reasoning.

A.1 Definition

Definition 1 (Core Spatial Concept Space)**.** *The spatial concept space is defined as:*

$$\mathcal{C} = \left\{ \begin{matrix} \textsc{Location}, \textsc{Object}, \textsc{Field}, \textsc{Event}, \\ \textsc{Network}, \textsc{Amount}, \textsc{Proportion} \end{matrix} \right\}. \tag{5}$$

Each concept $c \in \mathcal{C}$ represents a fundamental category of geographic phenomena and serves as an atomic unit in concept transformation reasoning.

A.2 Concept Semantics

Each core concept is associated with a distinct ontological interpretation:

- LOCATION: A spatial reference or place identifier, typically used to anchor other concepts (e.g., cities, regions, coordinates).
- OBJECT: Discrete spatial entities with identifiable boundaries (e.g., buildings, schools, fire stations).
- FIELD: Continuous spatial distributions over space (e.g., elevation, temperature, distance fields).
- EVENT: Temporally bounded spatial occurrences (e.g., traffic accidents, crime incidents).
- NETWORK: Graph-structured spatial systems supporting connectivity and routing (e.g., road networks, river networks).
- AMOUNT: Quantitative aggregations derived from objects, fields, or events (e.g., population count, total area).
- PROPORTION: Normalized quantities expressing ratios or densities (e.g., population density, percentage coverage).

A.3 Examples

Table 4 illustrates how natural-language phrases are mapped to core spatial concepts.

Natural-language phrase	Core concept
"fire stations"	OBJECT
"Euclidean distance"	FIELD
"traffic accidents in 2025"	EVENT
"road network"	NETWORK
"total population"	AMOUNT
"population density"	PROPORTION

Table 4: Examples of core spatial concepts extracted from natural-language questions.

B Functional Roles in Geo-Analytical Questions

To explicitly represent the procedural structure embedded in geo-analytical questions, we adopt a set of functional roles inspired by the GeoAnQu framework (Xu et al., 2023).

B.1 Definition

Definition 2 (Functional Role Set)**.** *We define the functional role space as:*

$$\mathcal{R} = \left\{ \begin{matrix} \textsc{Extent}, \textsc{TExtent}, \textsc{SubCond}, \\ \textsc{Cond}, \textsc{Support}, \textsc{Measure} \end{matrix} \right\}. \tag{6}$$

Each role $r \in \mathcal{R}$ specifies how a concept participates in a geo-analytical workflow.

B.2 Contextual vs. Procedural Roles

Functional roles are divided into two categories:

Contextual Roles

- EXTENT: Spatial scope restricting data collection (e.g., "in New York City").
- TEXTENT: Temporal scope restricting data collection (e.g., "in 2025").

Contextual roles constrain the domain of analysis but do not participate in procedural ordering.

Procedural Roles

- SUBCOND: Preliminary constraints that restrict candidate entities (e.g., "within 500 meters of rivers").
- COND: Conditions that further filter or transform intermediate results.
- SUPPORT: Spatial structures or reference objects used to compute derived concepts (e.g., road networks, buffers).
- MEASURE: The target output of the analysis.

B.3 Procedural Precedence

Procedural roles follow a strict execution order defined as:

$$\textsc{SubCond} \prec \textsc{Cond} \prec \textsc{Support} \prec \textsc{Measure}. \tag{7}$$

This ordering reflects the semantics of geo-analytical workflows: constraints are applied before supports are constructed, and measurements are computed last.

B.4 Examples

Table 5 shows functional role assignments for a representative geo-analytical question: "How many restaurants within 500m of coffee shops near parks along subway lines in San Francisco opened last week?"

Phrase	Concept	Role
"parks"	OBJECT	SUBCOND
"within 500m of coffee shops"	OBJECT	COND
"subway lines"	NETWORK	SUPPORT
"restaurants"	OBJECT	MEASURE
"in San Francisco"	LOCATION	EXTENT
"last week"	EVENT	TEXTENT

Table 5: Examples of functional role assignments in a geo-analytical question.

C Operator Library

We maintain a library of atomic operators spanning several functional categories. We detail representative operators below.

C.1 Geocoding and Location Resolution

geocode(t, a, r) Converts a textual address or place name t into geographic coordinates (ϕ, λ). An optional anchor location a provides disambiguation bias when multiple candidates exist. The region hint r (ISO 3166-1 alpha-2 code) further constrains the search space. When the primary geocoding API fails, a progressive fallback strategy employs nearby search with expanding radii (10km $\rightarrow$ 50km $\rightarrow$ 100km).

batch_geocode(T, a) Batch variant that processes a list of place names $T = \{t_1, \ldots, t_n\}$ with shared anchor bias, returning a list of resolved locations.

reverse_geocode(ϕ, λ) Inverse operation that converts coordinates to a human-readable address string.

C.2 Place Search and Retrieval

place_search(c, ρ, τ, κ, $r_{\min}$) Performs a spatial search centered at location c within radius ρ meters. Optional filters include place type τ (e.g., `restaurant`, `hospital`), keyword κ, minimum rating threshold $r_{\min}$, and current operating status.

place_details(p) Retrieves comprehensive metadata for place p, including name, coordinates, rating, price level, opening hours (structured as weekly periods), phone number, and associated place types.

batch_place_details(P) Batch variant that enriches a list of places P with detailed metadata, merging results with existing attributes.

C.3 Routing and Navigation

directions(o, d, m, W) Computes a route from origin o to destination d using travel mode $m \in \{\texttt{driving}, \texttt{walking}, \texttt{transit}, \texttt{bicycling}\}$. Optional waypoints $W = \{w_1, \ldots, w_k\}$ are automatically geocoded if provided as place names. Returns structured route data including legs, steps, distance, and duration. A waypoint verification mechanism checks whether specified intermediate stops are reflected in the returned route.

distance_matrix(O, D, m) Computes a $|O| \times |D|$ matrix of travel distances and durations between origin set O and destination set D using travel mode m.

compare_routes(R, μ) Compares multiple candidate routes $R = \{r_1, \ldots, r_n\}$ and returns the index of the optimal route according to metric $\mu \in \{\texttt{distance}, \texttt{duration}\}$.

filter_routes(R, κ) Filters routes based on instruction content, returning the index of routes containing (or avoiding) specific features indicated by keyword κ (e.g., `stairs`, `toll`, `roundabout`).

extract_distance(r) **/ extract_duration**(r) Utility operators that extract aggregate distance (meters) or duration (seconds) from a route object r.

C.4 Geometric Computation

haversine(ϕ_1, λ_1, ϕ_2, λ_2) Computes the great-circle distance between two points on Earth's surface using the Haversine formula:

$$d = 2R \cdot \arctan 2\left(\sqrt{a}, \sqrt{1-a}\right) \tag{8}$$

where $a = \sin^2\left(\frac{\Delta\phi}{2}\right) + \cos(\phi_1)\cos(\phi_2)\sin^2\left(\frac{\Delta\lambda}{2}\right)$ and $R = 6371$ km is Earth's mean radius.

bearing(ϕ_1, λ_1, ϕ_2, λ_2) Computes the initial bearing (forward azimuth) from point 1 to point 2, returned as degrees clockwise from true north (0°–360°).

bearing_to_direction(θ) Converts a numeric bearing θ to a cardinal/intercardinal direction string from the set {N, NE, E, SE, S, SW, W, NW}.

C.5 Spatial Analysis

nearest(a, C, μ) Finds the candidate $c^* \in C$ that minimizes distance to anchor point a. The metric μ can be `haversine` (geodesic distance) or `travel_time` (network-based).

within_radius(c, ρ, C) Filters candidates C to return only those within radius ρ meters of center point c.

pairwise_extremes(L) Identifies the pair of locations $(l_i, l_j) \in L \times L$ with maximum mutual distance, useful for determining spatial extent.

filter_places(P, θ) Filters a place list P according to constraints θ, which may include minimum rating, price level, required types, and operating status.

C.6 Temporal Reasoning

open_at_time(p, t) Determines whether place p is open at local datetime t by parsing structured opening hours. Handles edge cases including cross-midnight periods (e.g., 23:00–02:00) and 24-hour establishments.

timezone(ϕ, λ, τ) Retrieves timezone information for coordinates (ϕ, λ) at Unix timestamp τ, returning timezone ID, name, and UTC offset.

calculate_finish_time(t_0, L, S, m) Computes the finish time of a multi-stop itinerary starting at time t_0, visiting locations L with stay durations S using travel mode m. Automatically queries travel times between consecutive stops.

C.7 Trip Optimization

tsp_tw(D, L, S, W, t_0, T) Solves the Traveling Salesman Problem with Time Windows (TSP-TW) using Google OR-Tools. Given distance matrix D, locations L, service times S, time windows W, start time t_0, and time budget T, returns an optimized visit sequence that minimizes total travel time while respecting constraints.

When the complete solution violates time constraints, a greedy fallback algorithm constructs a partial feasible solution by iteratively adding the nearest unvisited location until the budget is exhausted.

steps_analysis(r, ℓ) Analyzes route instructions to extract navigation statistics: counts of left/right turns, roundabout exits, and optionally the instruction immediately following a specified landmark ℓ.

C.8 Local Context Operators

For efficiency, we maintain a local context database that caches frequently accessed data. Six operators provide database-first retrieval with API fallback:

- **query_local_place**: Retrieves place information from cache
- **query_local_coordinates**: Retrieves coordinates with geocode fallback
- **query_local_routes**: Retrieves cached route summaries
- **query_local_travel_time**: Retrieves cached travel time estimates
- **query_local_places_batch**: Batch place information retrieval
- **query_local_nearby_places**: Retrieves cached nearby search results

D Fine-tuning Details

D.1 Stage 1: Supervised Fine-Tuning

Given a training set $\mathcal{D}_{\text{SFT}} = \{(q_i, V_i)\}_{i=1}^{N}$ of question-concept pairs, we minimize the negative log-likelihood:

$$\mathcal{L}_{\text{SFT}} = -\sum_{i=1}^{N} \log p_\theta(V_i \mid q_i) \tag{9}$$

where θ denotes the model parameters, and V_i represents the annotated concepts with their types from $\mathcal{C}$ and functional roles from $\mathcal{R}$.

D.2 Stage 2: Direct Preference Optimization

For each question q, we construct preference pairs (G^+, G^-) where $G^+ \in \bigcap_{i=1}^{5} \mathcal{G}_i$ satisfies all well-formedness constraints, while $G^- \notin \bigcap_{i=1}^{5} \mathcal{G}_i$ violates at least one. The DPO objective is:

$$\mathcal{L}_{\text{DPO}} = -\mathbb{E}_{(q,G^+,G^-)}\left[\log \sigma\left(\beta \cdot r_\theta(q, G^+, G^-)\right)\right] \tag{10}$$

where $r_\theta(q, G^+, G^-) = \log \frac{p_\theta(G^+|q)}{p_{\text{ref}}(G^+|q)} - \log \frac{p_\theta(G^-|q)}{p_{\text{ref}}(G^-|q)}$, σ is the sigmoid function, β is the temperature parameter, and p_{ref} is the reference model (initialized from Stage 1). This formulation encourages the model to prefer graphs satisfying $\mathcal{G}_1$ (acyclicity), $\mathcal{G}_2$ (role ordering), $\mathcal{G}_3$ (type compatibility), $\mathcal{G}_4$ (data availability), and $\mathcal{G}_5$ (connectivity).

E Template Library

Our system employs a library of macro-templates that cover the major spatial reasoning patterns:

FILTER-AGGREGATE-MEASURE Filters objects by spatial/temporal predicates, aggregates results, and computes a final measure.

OBJECT-FIELD-MEASURE Transforms discrete locations into continuous fields (distance, bearing) for measurement.

ROUTE-OPTIMIZE Computes optimal visiting order for multi-stop trips with time windows.

GEOCODE-BATCH-COMPARE Batch geocodes candidates and compares against a reference point.

LOCATION-BEARING-CLASSIFY Computes bearing angle and converts to cardinal direction.

ROUTE-STEP-EXTRACT Analyzes navigation steps for specific maneuvers.

MULTI-ROUTE-COMPARE Computes multiple candidate routes and selects by metric.

PLACE-ATTRIBUTE-QUERY Retrieves POI details and queries temporal/rating attributes.

MULTI-SEGMENT-AGGREGATE Computes multi-leg journeys with mixed transport modes.

TIME-WINDOW-REVERSE Reverse-calculates latest departure given a deadline constraint.

E.1 Example Retrieval Mechanism

To provide semantic guidance during compositional generation, we maintain an example store $\mathcal{E} = \{(q_i, G_i)\}_{i=1}^N$ of question-graph pairs. Each graph G_i is serialized into a textual description. Given an input question q, we retrieve the top-k most similar examples by computing cosine similarity between the embedding of q and each stored question q_i. The retrieved examples guide parameter binding and edge instantiation during template-based composition.

Algorithm 1 GeoFlow Graph Execution

Require: Factorized graph $G' = (V', E')$, initial state Σ_0
Ensure: Final state Σ_M, execution trace $\mathcal{F}$

```
T ← TopologicalSort(G')        ▷ Order by role priority
Σ ← Σ_0; F ← []
for each step s_i = (B_i, A_i, ω_i, θ_i) ∈ T do
    inputs ← {Σ(v) : v ∈ B_i}
    outputs ← ω_i(inputs; θ_i)
    for each v_j ∈ A_i do
        Σ(v_j) ← outputs[j]
    end for
    F.append((s_i, Σ))
end for
return Σ, F
```

F Execution and Response Generation Details

F.1 Execution Semantics

Let $\Sigma : V \to \mathcal{D}$ denote the concept state, mapping each entity to its resolved data value in domain $\mathcal{D}$. Starting from an initial state Σ_0 (with extent nodes grounded to input data), we iteratively apply each operator to update the state until all nodes are resolved.

The execution trace $\mathcal{F} = [(s_1, \Sigma_1), (s_2, \Sigma_2), \ldots, (s_M, \Sigma_M)]$ records the state snapshot after each operator execution. For example, given a place name "Central Park", the first operator `geocode` resolves it to coordinates $(40.78, -73.97)$, updating Σ_1; a subsequent `haversine` operator then computes the distance to another location, yielding Σ_2 with the distance value. This trace enables interpretability and debugging. The complete procedure is given in Algorithm 1.

F.2 Grounded Response Generation

The executed GeoFlow Graph yields a final state Σ_M containing structured geospatial evidence (e.g., computed distances, optimized routes, filtered place rankings, and temporal constraints) derived through the agent's tool invocations and spatial computations. The agent generates its final response by grounding on these verifiable intermedi-

ate results produced during graph execution.

Formally, the agent synthesizes the final response by conditioning on both the original question and the execution outcomes:

$$a = f_{\text{gen}}(q, \Sigma_M, \mathcal{F}) \tag{11}$$

where $\mathcal{F}$ denotes the execution trace providing step-by-step reasoning evidence. This formulation enables *grounded generation*: the response is factually anchored in computational results obtained through tool-augmented reasoning, rather than hallucinated from learned priors.