Geo-ReAct: An Agentic AI Framework with Concept-Grounded Tools for Geospatial Question Answering

Zabir Al Nazi
University of California, Riverside
Riverside, USA
znazi002@ucr.edu

Zhuocheng Shang
University of California, Riverside
Riverside, USA
zshan011@ucr.edu

Shahd Elmahallawy
University of California, Riverside
Riverside, USA
selma008@ucr.edu

Vagelis Hristidis
University of California, Riverside
Riverside, USA
vagelis@cs.ucr.edu

Ahmed Eldawy
University of California, Riverside
Riverside, USA
eldawy@ucr.edu

Abstract

Geospatial question answering (QA) is critical for many important applications. While large language models (LLMs) can adequately handle the linguistic and reasoning requirements, they lack reliable mechanisms for the geometric operations that complex geospatial questions demand. That is, geospatial QA requires combining the language and reasoning capabilities of LLMs with the expertise of geospatial data management platforms. We present Geo-ReAct, an agentic framework that addresses this by equipping an LLM with geospatial tools grounded in Kuhn's core concepts of spatial information. These specialized tools overcome the limitations of general-purpose agentic platforms, as shown through our experiments. The LLM in Geo-ReAct handles language understanding and multi-step planning while its specialized tools handle each spatial subtask: question parsing, schema inspection, location grounding, neighborhood construction, metric aggregation, and query execution with iterative refinement. We performed experiments on two comprehensive geospatial QA datasets, GS-QA (2,800 questions, 28 templates) and MapQA (2,206 questions, 9 reasoning types). Geo-ReAct achieves 62.5% and 51.9% entity accuracy, and 69.8% and 88.8% numeric accuracy, respectively. This outperforms the strongest baseline by up to 14.8 and 60.8 percentage points, respectively. The most significant gains occur on questions requiring spatial metric computation, directional reasoning, and spatial aggregation, where the absence of dedicated spatial tools proves most limiting.[1]

Keywords

Geospatial Question Answering, Agentic AI, OpenStreetMap, Tool-augmented Reasoning, Large Language Models

[1]The source-code for Geo-ReAct is available here: https://anonymous.4open.science/r/geo-react/

Conference'17, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YYYY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

ACM Reference Format:
Zabir Al Nazi, Zhuocheng Shang, Shahd Elmahallawy, Vagelis Hristidis, and Ahmed Eldawy. 2026. Geo-ReAct: An Agentic AI Framework with Concept-Grounded Tools for Geospatial Question Answering. In . ACM, New York, NY, USA, 14 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Geospatial reasoning is critical in many applications such as urban planning, transportation, environmental monitoring, and public health [20, 38]. Answering such questions at scale requires structured geospatial knowledge bases that can be used to execute spatial predicates correctly and a reasoning architecture that can coordinate them. OpenStreetMap (OSM) [7], which contains over 11 billion spatial objects, is the world's largest open geospatial database.

A key limitation has been that extracting answers from these resources requires expertise in spatial query languages and GIS conventions. Large language models offer a promising path toward lowering this barrier: recent studies demonstrate that LLMs encode substantial geographic knowledge and can reason about spatial language, including relative directions, proximity, and containment [18, 22].

However, despite the rapid advances in LLMs, current spatial reasoning systems are ineffective in facilitating fluent linguistic understanding matched with accurate spatial computation [40]. Geospatial questions present unique challenges for automated question answering (QA). Unlike factoid queries that retrieve discrete facts, spatial questions involve multiple interacting constraints: semantic types must be mapped to schema representations, reference locations must be grounded to coordinates, and spatial relations such as distance, direction, and topology must be translated into geometric predicates. Consider the question: "What is the total area of public parks within 25 kilometers of downtown Los Angeles that lie northeast of LAX?" Answering it requires resolving two named locations—downtown Los Angeles and LAX—into spatial references, evaluating the 25 km constraint as a geodesic distance, computing the azimuth from LAX to each candidate park, and keeping only those in the northeast quadrant, selecting the OSM relation whose geometry supports area, and summing the per-park areas. The choice of geometry representation also matters: the same real-world feature class – such as a park – may be stored as both a point and a polygon in a spatial database, and an area computation

over a point geometry produces zero rather than an error, silently returning a wrong answer.

Several other types of errors may occur, such as resolving an ambiguous place name to the wrong location, or applying a distance predicate in planar rather than geodesic coordinates. Because each stage depends on the outputs of the preceding ones, a single incorrect decision propagates through the pipeline undetected. A system that performs correctly on simple lookups can therefore produce semantically or spatially incorrect answers on questions requiring chained spatial predicates [12, 19].

Another suite of challenges stems from the nature and complexity of geospatial data collections. OSM contains 107,000 distinct tag keys, and a free-form schema where contributors can attach arbitrary key-value pairs to any entity, querying OSM demands expertise in spatial databases, PostGIS functions, and platform-specific tagging conventions – compounding the human and LLM expertise barrier that limits access to the geospatial knowledge.

To overcome these challenges, this paper introduces Geo-ReAct, an agentic framework for spatial QA, where the core language and reasoning competencies of an LLM are augmented by specialized geospatial tools. This approach is supported by recent work on Agentic AI, i.e., on tool-augmented LLMs, that demonstrates that LLMs can effectively delegate specialized computation to external tools while retaining control over multi-step reasoning and planning [29, 39]. That is, the LLM handles language understanding and reasoning while specialized tools perform spatial computation.

A key finding of this paper is that a general-purpose agentic architecture – consisting of an LLM and data access tools, such as SQL querying – fails to correctly answer many spatial questions. Specifically, general-purpose tool-augmented agents equipped with SQL execution can recover from query errors interactively [6], but without tools that encode spatial semantics, they must rediscover coordinate conventions, PostGIS function signatures, and schema mappings at every invocation.

On the other extreme, wrapping individual PostGIS functions as tools produces an interface that is too large to be useful: agents equipped with excessive tool inventories are known to underperform due to tool selection overhead, unnecessary exploration, and difficulty identifying the relevant action among many candidates [11, 17]. *The central design question is therefore at what level of abstraction to define the tool interface* - broad enough to encode meaningful spatial semantics, narrow enough to keep the action space tractable for compositional reasoning.

Kuhn's Core Concepts of Spatial Information [15] offer a theoretical foundation to create spatial tools at the right level of abstraction. Kuhn identified ten concepts (Location, Neighborhood, Field, Object, Network, Event, Granularity, Accuracy, Meaning, and Value) that capture how humans reason about space, explicitly designed to enable non-specialists to work with spatial information without requiring GIS expertise. Subsequent work has demonstrated that these concepts can systematically bridge natural language and computational GIS operations [16, 28]. LLMs align naturally with this framework: they can reason about locations, neighborhoods, and spatial relations in language while delegating geometric computation to concept-aligned tools that implement the corresponding database operations.

We present Geo-ReAct, a concept-grounded agentic framework over PostgreSQL/PostGIS that implements eight tools: three general tools for schema inspection, SQL generation, and iterative query refinement and five spatial tools that realize four of Kuhn's concepts (Location, Neighborhood, Object, Field) as executable operations. Individual spatial tools serve multiple modes, so four concepts yield five tools organized into grounding, computation, and execution functions. Of the six remaining Kuhn concepts, three (Granularity, Accuracy, Meaning) are second-order qualities that refine how primary concepts are computed, and three (Network, Event, Value) require data structures absent from standard OSM imports or fall outside factoid QA scope. This concept-level granularity encodes spatial semantics that LLMs cannot reliably reconstruct while keeping the action space tractable for compositional reasoning.

To evaluate Geo-ReAct, we performed comprehensive experiments on two complementary geospatial QA benchmarks: GS-QA [27], a benchmark comprising 2,800 questions across 28 templates of high spatial complexity over a United States OSM database; and MapQA [19] (2,206 questions, 9 reasoning types) over Southern California OSM, which introduces additional spatial reasoning patterns including adjacency, road-junction queries, and pairwise comparison. Together, the two benchmarks cover a broad range of spatial question types and database scales.

Using Gemini-2.5-Flash as the backbone LLM for all systems, our experiments show that Geo-ReAct substantially outperforms multi-agent SQL pipeline, text-to-SQL and state-of-the-art agentic ReAct-style baselines. We found that the largest gains occur on questions requiring directional reasoning, multi-anchor constraints, and spatial aggregation. Geo-ReAct also produces executable queries more reliably than all baselines, achieving 93.5% and 98.6% valid execution rates on GS-QA and MapQA respectively, and our tool utility analysis confirms that each spatial tool activates predominantly on questions involving its corresponding spatial concept, with the highest activation rates coinciding with the largest accuracy gains over baselines. In addition to an increase in accuracy, we show that Geo-ReAct achieves significant cost savings compared to general-purpose ReAct agents.

Our key contributions are as follows:

- A principled methodology for geospatial agent tool design grounded in Kuhn's core concepts of spatial information.
- Geo-ReAct, a concept-grounded agent with tools for location grounding, schema resolution, spatial transformation, and query refinement over PostgreSQL/PostGIS databases.
- Evaluation on the GS-QA and MapQA datasets, demonstrating significant improvements over baselines while requiring fewer tokens and lower latency than comparable agentic systems.

2 Related Work and Background

Geospatial question answering requires bridging natural language understanding with spatial computation, a challenge that prior work has approached through RAG-style query generation, knowledge graph construction, tool-augmented reasoning, and autonomous GIS workflows. We review these approaches below and identify the gaps that motivate Geo-ReAct for agentic geospatial question answering over OpenStreetMap.

2.1 Geospatial Question Answering

Early geospatial QA systems relied on template-based parsing to translate natural language into structured queries over linked data. GeoQA [24] introduced this approach over DBpedia and OSM using GeoSPARQL, and subsequent systems including GeoQuestions1089 [13] and GeoQA2 [14] extended the paradigm to YAGO2geo with broader spatial coverage. Text-to-OverpassQL [33] targets live OSM querying through the Overpass API, supporting a broader set of question types but remaining limited by the API's lack of full SQL semantics such as joins and aggregations.

Recent benchmarks expose persistent limitations. MapQA [19] demonstrates that text-to-SQL approaches collapse on multi-hop spatial reasoning, particularly on queries requiring intersection and distance predicates, where answers require chaining multiple spatial operations. MapEval [4] evaluates foundation models on 700 map-based questions and finds that no model exceeds 67% accuracy. The GS-QA benchmark [27] introduces 2,800 questions across 28 templates that combine distance, direction, topology, and aggregation constraints, further highlighting the difficulty of compositional spatial queries. These results collectively point toward a shared conclusion: architectures that attempt to generate spatial queries in a single pass are fundamentally limited by the compositional complexity of geospatial questions.

2.2 Geospatial Knowledge Bases and Query Interfaces

OpenStreetMap is the world's largest open geospatial database, containing over 11 billion spatial objects with more than 107,000 distinct tag keys. Despite this scale and richness, most geospatial QA systems access OSM indirectly. Knowledge graph extractions such as WorldKG [5] and YAGO2geo [10] provide structured access but discard OSM's free-form tagging scheme in the process. The Overpass API [33] preserves more of the original schema but lacks support for complex spatial joins and aggregations that SQL provides. External map services introduce latency, rate limits, and coverage gaps.

Knowledge graphs impose fixed ontologies on a schema-free dataset. API-based approaches cannot express the complex spatial joins needed for multi-constraint questions, such as filtering by distance, direction, and semantic type simultaneously. Geo-ReAct operates directly over a PostgreSQL/PostGIS instance of OSM, retaining the full relational schema and supporting complex spatial joins and aggregations without intermediary abstractions, and enabling deterministic, reproducible spatial query execution.

2.3 Spatial Reasoning in Large Language Models

A growing body of work investigates the spatial capabilities of LLMs and consistently finds a gap between qualitative understanding and quantitative computation. GeoLLM [22] shows that augmenting prompts with OSM contextual information improves socioeconomic predictions, demonstrating that LLMs can leverage spatial context when it is provided. GPT4GEO [26] documents that LLMs produce plausible qualitative spatial descriptions but generate quantitatively inaccurate distances and measurements. On textual reasoning benchmarks, StepGame [30] and subsequent evaluations [18] show that accuracy decays as the number of spatial reasoning hops increases.

Collectively, these results indicate that - LLMs can interpret spatial language, understand that "within 50km towards the airport" combines distance and directional constraints, and plan multi-step reasoning strategies. However, they cannot reliably compute geometric operations such as coordinate transformations, azimuth calculations, or topological predicates. This asymmetry between spatial understanding and spatial computation motivates the delegation of geo-spatial operations to external tools.

2.4 Tool-Augmented Language Models

ReAct [39] established the standard paradigm for tool-augmented reasoning by interleaving thought traces with tool calls in a unified action-observation loop, enabling LLMs to reason about when and how to use external tools. Toolformer [29] demonstrated that LLMs can learn to invoke tools through self-supervision, and Reflexion [31] added self-critique mechanisms that allow agents to refine their strategies across attempts. ToolLLM [25] scaled tool use to over 16,000 real-world APIs.

These frameworks demonstrate that LLMs can reliably externalize computation through well-designed tool interfaces. However, these generalist systems do not include tools with spatial semantics. A general-purpose ReAct agent equipped with a SQL executor can generate and refine queries [35], but without tools that encode spatial concepts, it must rediscover coordinate conventions, PostGIS function signatures, and tagging schemes on invocation for every question.

2.5 Autonomous GIS Agents

Recent systems apply LLM agents specifically to geospatial tasks. Autonomous GIS [20] proposed agents that compile solution graphs into executable Python for spatial analysis. GIS Copilot [1] orchestrates over 100 QGIS tools for desktop GIS workflows, and GeoGPT [42] integrates OSM and remote-sensing tools for geographic analysis. These systems produce analysis workflows (e.g., selecting counties exceeding a population threshold, mapping the spatial distribution of COVID-19 cases across administrative units, or downloading and joining boundary and demographic data for a given region) rather than direct answers to natural-language questions; their design targets GIS practitioners performing exploratory spatial analysis, not end-users seeking factual answers from geospatial databases.

Spatial-Agent [2] pre-compiles questions into static GeoFlow Graphs (directed acyclic graphs) before execution and routes computation through external geospatial APIs. Its error analysis attributes the majority of failures to API-level data quality issues. Geo-ReAct instead operates through a ReAct loop that reasons step by step and adapts dynamically when intermediate results are unexpected. Rather than relying on a template library that must be expanded for novel question types, Geo-ReAct exposes Kuhn's concepts as composable tools that the agent chains freely based on the structure of each question. Rather than relying on external APIs,

Geo-ReAct executes queries directly against a local PostGIS database, eliminating API variability and enabling the complex spatial joins that API-based approaches cannot express.

Existing geospatial QA systems fall into one of several categories, each with characteristic limitations. Direct query generation approaches (text-to-SQL, text-to-OverpassQL) fail on compositional spatial predicates because they must produce a complete, correct query in a single pass. Systems that access OSM through knowledge graphs or APIs lose the expressive power needed for complex spatial joins. Autonomous GIS agents focus on analysis workflows rather than question answering. General-purpose tool-augmented agents lack spatial priors and must rediscover domain knowledge at inference time.

Geo-ReAct bridges these gaps through concept-aligned tools over a PostgreSQL/PostGIS OSM database, combined with adaptive ReAct reasoning. The tools encode the domain knowledge that the LLM lacks (coordinate grounding, schema resolution, spatial predicates), while the LLM provides the compositional reasoning that rigid pipelines cannot match. This combination enables the agent to handle the full range of spatial complexity in geospatial QA benchmarks while maintaining the flexibility to adapt when intermediate results are unexpected.

2.6 Background on Core Concepts of Spatial Information

Kuhn [15] proposed ten core concepts of spatial information, explicitly intended to enable non-specialists to work with spatial data. The ten concepts (Location, Neighborhood, Object, Field, Network, Event, Granularity, Accuracy, Meaning, and Value) capture how humans reason about geographic phenomena without requiring expertise in GIS software or spatial query languages. Kuhn and Ballatore [16] subsequently operationalized these concepts as an executable language for spatial computing, and Scheider et al. [28] formalized them as a type system for geo-analytical question answering, demonstrating that the concepts compose into well-typed analytical workflows [2].

We observe that this framework aligns naturally with the capabilities and limitations of LLM agents. The concepts describe spatial reasoning at the level of human cognition: locations have names, neighborhoods define proximity, objects carry semantic types, and spatial measurements reflect continuous fields. An LLM already reasons fluently at this level. What it lacks is the ability to execute the corresponding geo-spatial operations. By integrating the concepts with the tools, we create an interface where the LLM's natural reasoning maps directly to correct spatial computation, without requiring the model to encode knowledge of PostGIS syntax or OSM tagging conventions.

3 Problem Formulation

Let $\mathcal{D} = \{R_1, R_2, \ldots, R_n\}$ be a geospatial database where each relation R_i contains tuples with a geometry attribute g_i storing Point, LineString, or Polygon geometries (a single relation may contain mixed types). $\mathcal{D}$ is stored in a geospatial database system, such as PostgreSQL with PostGIS.

Given a natural language question q, the geospatial question answering task is to produce a correct answer to q by constructing and executing a SQL query over $\mathcal{D}$. The query may combine standard relational operators (joins, filters, aggregations, grouping) with PostGIS spatial functions: proximity predicates (`ST_DWithin`, `ST_Distance`), topological predicates (`ST_Intersects`, `ST_Contains`), geometric computations (`ST_Area`, `ST_Length`, `ST_Azimuth`), and nearest-neighbor ordering. In the generation of that query, additional tools can be used, e.g., location resolution, to produce the final query. Note that multiple SQL queries can be generally combined into one SQL query so this does not limit the expressiveness of the system. A single question typically requires composing several such operators: semantic types must be mapped to schema filters, reference locations must be grounded to coordinates, spatial relations must be translated into geometric predicates, and the results may be projected, aggregated, or ranked. Some questions additionally require retrieval from external knowledge sources $\mathcal{E}$ beyond $\mathcal{D}$ (e.g., encyclopedic attributes of matched entities), yielding multi-hop reasoning.

Constructing such queries requires coordinating multiple interdependent steps: grounding named locations to coordinates, mapping semantic types to schema-level filters, selecting geometry representations that match the requested operation, and composing these into correct SQL. We model this as a *sequential decision process*: an agent selects tools $t_1, t_2, \ldots, t_k$ where each t_i resolves one or more subtasks conditioned on prior observations $o_1, \ldots, o_{i-1}$. When o_i signals failure, the agent revises earlier resolutions by selecting corrective tools. For example, given "How many parks are within 25 km of downtown Los Angeles?", the agent may select ResolveLocation["downtown Los Angeles"] $\rightarrow$ (*lat*, *lon*), then GetPOIFilter[{park, location}] $\rightarrow$ (*table*, *column*, *value*, *geom-type*), then either CalculateMetric to compute the count or GenerateSQL to compose a `ST_DWithin` query, depending on the spatial constraints involved.

4 Geo-ReAct

Geo-ReAct implements the sequential decision process of Section 3 using the ReAct framework [39]. At each step i, the agent generates a natural-language thought τ_i explaining its reasoning, selects a tool $t_i \in \mathcal{T}$ with arguments, receives the tool's output o_i (e.g., resolved coordinates, a schema filter, or SQL query results), and appends (τ_i, t_i, o_i) to the trajectory. The process terminates when the agent issues a Finish action or reaches K iterations (we set $K = 10$ in our experiments). A key design decision is the composition of $\mathcal{T}$, discussed in Section 4.1.

4.1 Principled Spatial Tool Design

An LLM can plan over spatial constraints that it understands linguistically, but executing the geometric operations that those constraints demand requires external tools. Regarding tool selection, the one extreme is to only use the standard agentic tools employed in database-backed applications, including general-purpose NL2SQL [36]: SQL generation and schema inspection. This pushes to the agent all specialized spatial reasoning and computation. The other extreme is wrapping each individual PostGIS function as a tool. This can lead to high selection overhead and unnecessary

Table 1: The eight Geo-ReAct tools. Role: G=grounding, C=computation, E=execution. Kuhn concepts: Loc=Location, Nbh=Neighborhood, Obj=Object, Fld=Field; ∗=second-order (A=Accuracy, G=Granularity, M=Meaning).

	Tool	Role	Description	Input (example)	Output (example)	Kuhn Concepts
Spatial Tools	ParseQuestion	G	Extract structured spatial parameters via constrained LLM call	NL question (*"parks within 25 km of downtown LA"*)	{feature_type, metric, anchors, direction, radius, region}	M/G*
	ResolveLocation	G	Map a place name or entity ID to a point coordinate	Location string (*"downtown Los Angeles"*)	(lat, lon)	Loc
	GetRegionGeometry	G	Retrieve admin boundary polygon, radius buffer, or street intersection	Region, state (*"Douglas County", "NE"*)	Region polygon or buffer predicate	Loc, Nbh
	GetPOIFilter	G	Resolve feature type to schema filter, indexed by geometry type	{poi_type, query_type} (*{"park", "area"}*)	(table, column, value, geom-type)	Obj
	CalculateMetric	C	Compute spatial metric (count, area, length, distance, directional-nearest)	{metric, table, filter_col, filter_val, region_name, region_state}	Aggregate value or nearest entity	Obj, Fld
General Tools	GenerateSQL	E	Preprocess and execute an agent-generated PostGIS query	SQL string	Query results or error	Substrate
	RefineSQL	E	Revise and re-execute previous query given error or empty result	Revised SQL string	Query results or error	Substrate
	InspectSchema	G	Return column statistics (null fraction, distinct count, common values)	{table, columns} (*{"pois", ["cuisine"]}*)	Column statistics	A/G/M*

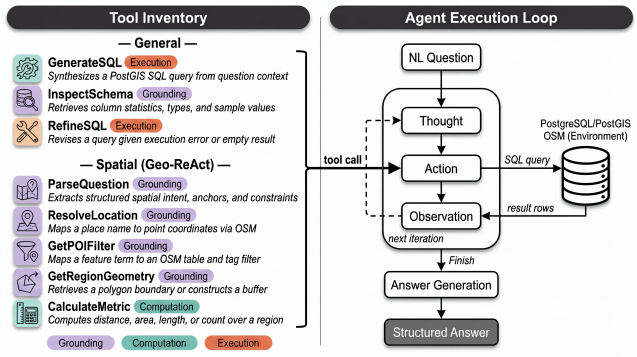


Figure 1: Geo-ReAct architecture. The agent operates within a ReAct loop over a PostgreSQL/PostGIS OSM database.

exploration [11, 17]. The central design question is at what level of abstraction to define the tool interface.

We resolve this by aligning tools with Kuhn's [15] core concepts of spatial information, which were designed to let non-specialists reason about geographic data without GIS expertise. We argue that this is the level of abstraction needed by a modern agent, which reasons fluently about spatial language but cannot reliably execute geometric operations.

Geo-ReAct implements eight tools, split into two groups, as shown in Table 1.

Each tool serves one of three roles: *grounding* (resolving unstructured input to a structured reference), *computation* (evaluating a spatial predicate or metric), or *execution* (producing and running SQL). Table 1 summarizes all eight tools with the concept mapping.

4.1.1 General Tools. GenerateSQL accepts an agent-generated PostGIS query, preprocesses it, and executes it against the database. RefineSQL does the same for a revised query issued after an execution error or an empty result set. Both share the same execution path: the query passes through a deterministic preprocessor, is validated for read-only safety, and is run against the database with a timeout. The preprocessor normalizes common LLM-generated SQL artifacts (markdown fences, operator typos, trailing commas) and repairs PostGIS dialect mismatches using a declarative type-requirement table that maps PostGIS functions to their required input types (`geometry` or `geography`), inserting correct casts automatically. Each applied fix is logged in the observation so the agent can reason about what changed.

InspectSchema takes a table name and an optional list of columns as input, queries the PostgreSQL catalog, and returns their fraction of null values, estimated distinct count, and most-common values. These signals help the agent determine which columns are informative and what filter values to use before committing to a query.

4.1.2 Kuhn-inspired Spatial Tools. ParseQuestion extracts structured spatial parameters from the natural-language question. It issues a single bounded call to the LLM with a constrained extraction prompt and a fixed output schema (feature type, metric, anchor locations, direction, radius, region, and schema-specific fields such as street names for MapQA). Unlike the agent's open-ended planning calls, this extraction step has a well-defined output type and does not participate in the ReAct loop's iterative reasoning.

ResolveLocation maps a place name or entity identifier to a point coordinate. It searches the relevant tables in priority order, applying available disambiguation filters (city, state, identifier type). On GS-QA, the search spans five thematic tables; on MapQA, it implements a multi-tier cascade across the point and polygon tables, handling both positive and negative identifiers and coordinate transforms. In both cases, polygon matches are reduced to their centroid.

GetRegionGeometry maps a spatial context description to a predicate that downstream queries use in their `WHERE` clause. It operates in three modes. The primary mode searches administrative boundaries across multiple levels, generating name variants and querying both name and metadata columns for matching. When multiple candidates match, a distance-based threshold relative to the target location disambiguates them. The output is a region polygon. When no administrative polygon exists, a fallback mode resolves

the place name and emits a radius buffer predicate. Additionally, the tool resolves street intersections: given two street names, it first attempts geometric intersection of the line features, then falls back to a closest-pair computation with a distance sanity check, returning a point coordinate.

GetPOIFilter resolves a natural-language feature type (e.g., "park") to (table, column, filter-value, geometry-type) tuples specifying a schema predicate: for example, "park" with query_type="area" yields (`parks`, `leisure`, `'park'`, `polygon`), i.e. `WHERE leisure = 'park'` over polygon geometries. The tool uses a two-tier strategy: a curated mapping from systematic schema profiling, with fallback dynamic discovery across categorical columns. Results are grouped by intended operation - polygon entries for area, line for length, point for count - so downstream tools receive the correct geometry type for the requested operation.

CalculateMetric takes a metric type, a feature filter (table, column, value), and a spatial constraint (region boundary, radius, cardinal direction, or their combination), and returns a numeric result with its unit (e.g., 450,000 m^2, 12.4 km, or count=7) or the nearest matching entity for directional-nearest queries. It supports five modes: *count*, *area*, and *length* aggregate over spatially scoped features; *distance* computes the geodesic distance between two entities; and *directional-nearest* finds the nearest entity in a cardinal direction by combining nearest-neighbor search with an azimuth filter.

5 Baselines

We consider the following baselines, which collectively cover the state-of-the-art in NL2SQL and agentic database-backed platforms. All baselines query the same PostgreSQL/PostGIS database with full access to spatial functions.

5.1 Text-to-SQL

The Text-to-SQL baseline follows the implementation of GS-QA [27] and serves as the canonical single-pass reference for LLM-driven SQL generation. Given a natural-language question, the pipeline executes sequential stages with no feedback between them:

(1) **SQL generation.** The question is passed to the LLM together with a static system prompt that contains the schema of all the OSM tables and a set of PostGIS conventions for spatial queries (geography casts for distance queries, `ST_DWithin` / `ST_Distance` for radius and proximity, `<->` for nearest-neighbor ordering, etc.). The LLM emits a single SQL query enclosed in a fenced code block.

(2) **Execution.** The generated query is run against PostGIS with a 120-second timeout; any execution error is recorded but not fed back to the model.

(3) **Answer generation.** The retrieved rows are appended to an answer-generation prompt, and the LLM produces a natural-language response constrained to the returned records.

It establishes how well a single LLM call can handle geospatial SQL given only the schema and PostGIS conventions without error feedback or schema exploration.

5.2 CHESS

CHESS [34] is a multi-agent text-to-SQL framework that decomposes query synthesis into four specialized agents. **1)** The *Information Retriever* extracts relevant entities and schema descriptions using keyword extraction, locality-sensitive hashing over database values, and embedding-based retrieval over the database catalog. **2)** The *Schema Selector* prunes the schema down to the tables and columns relevant to the question, which is essential for large schemas where overloading the prompt degrades generation quality. **3)** The *Candidate Generator* produces SQL candidates and iteratively revises them using execution feedback when queries fail or return empty results. **4)** The *Unit Tester* generates natural-language unit tests that distinguish candidates and selects the highest-scoring query as the final output. We include CHESS as a strong representative of structured multi-agent SQL synthesis.

5.3 ReAct Agent with Schema Exploration

We design a strong ReAct-style SQL agent following the design of RAISE [6], which augments interactive SQL agents with a dedicated schema-inspection tool. At each step the agent emits a *Thought* explaining its reasoning, takes an *Action* from a fixed action space, and receives an *Observation* from the environment. The action space consists of four tools:

- `GenerateSQL[query]` – produce an initial SQL query.
- `RefineSQL[query]` – revise the previous query given the latest observation or error.
- `InspectSchema[{table, columns}]` – probe a table for column existence, types, and example values, following the schema-exploration step introduced by RAISE.
- `Finish[answer]` – terminate and return the final answer.

The loop runs until the agent issues `Finish` or reaches a cap of ten iterations, after which the same answer-generation and JSON-parsing stages used by the Text-to-SQL baseline convert the retrieved rows into the structured output format. The system prompt provides the OSM schema and the PostGIS conventions for spatial queries. Compared to the single-pass baseline, the ReAct agent can recover from SQL errors and resolve schema ambiguity before committing to a final query. This baseline isolates the value of iterative, tool-augmented SQL exploration without the broader geo-spatial tool suite that our method introduces.

6 Experimental Setup

6.1 Benchmarks and Model

We first evaluate on **GS-QA** [27], a comprehensive open-domain geospatial QA benchmark, with 2,800 question–answer pairs generated from 28 templates over a PostGIS database built from OpenStreetMap covering the entire United States region. Its templates span seven spatial predicate combinations (nearest neighbor, range, direction, towards, intersects, and their combinations) and eight answer types (entity name, location, direction, distance, area, length, count, and multi-hop external attribute), making it well-suited for a thorough evaluation of geospatial QA systems. We then evaluate on **MapQA** [19], a benchmark with 2,206 question–answer pairs across nine reasoning types over OSM data for Southern California. Examples from each benchmark are shown in Table 3.

Table 2: Main results on GS-QA and MapQA. Best per column in bold.

System	GS-QA						MapQA					
	Accuracy @ τ		*Cost & Efficiency*				*Accuracy @ τ*		*Cost & Efficiency*			
	Ent. τ_{ent} ▲	**Num.** τ_{num} ▲	**VER** ▲	**LLM Calls** ▼	**Tokens (1k)** ▼	**Latency (s)** ▼	**Ent.** τ_{ent} ▲	**Num.** τ_{num} ▲	**VER** ▲	**LLM Calls** ▼	**Tokens (1k)** ▼	**Latency (s)** ▼
CHESS [34]	11.2	6.3	60.4	16.7	21.1	107.2	28.9	5.6	80.3	4.2	6.1	17.0
Text-to-SQL† [27]	17.8	19.8	30.6	**2.6**	**1.1**	**8.6**	39.6	8.4	62.9	**1.0**	**1.1**	**3.3**
RAISE (ReAct Agent) [6]	47.7	24.1	79.7	7.5	22.6	15.8	45.6	28.0	96.1	3.7	7.8	7.1
Geo-ReAct (ours)	**62.5**	**69.8**	**93.5**	6.1	10.7	10.5	**51.9**	**88.8**	**98.6**	3.1	6.9	6.6
Templates	T1–T22 (Entity); T23–T28 (Numeric)						Type 1–8 (Entity); Type 9 (Numeric)					

τ_{ent}=1.0 (token containment); τ_{num}=0.10 (10 % relative error); **VER**: valid execution rate - fraction of questions for which the system returns a non-empty answer without execution errors (%); **LLM Calls**: mean number of LLM inference invocations per question, aggregated over all pipeline stages; **Tokens**: mean total tokens per question, in thousands; **Latency**: mean wall-clock time per question, in seconds; † fixed-stage pipeline. All systems use Gemini 2.5 Flash. ▲ = higher is better; ▼ = lower is better.

We run all baselines and our method with **Gemini-2.5-Flash** [3] as the backbone LLM. Recent benchmarks on geospatial tasks show that Gemini-2.5-Flash delivers strong accuracy at substantially lower inference cost than comparable models [8, 37]. makes it a well-suited choice.

6.2 Evaluation Metrics

The answers in both benchmarks are either an *entity* (place or feature names) or a *numeric value* (count, distance, area, length). We report a unified **accuracy@τ**: a result is correct if it falls within a type-specific tolerance τ of the ground truth, and we average this binary signal within and across templates.

Entity answers. Let $T(\hat{e})$ and $T(e)$ be the token sets of the *answer* and *ground truth*, respectively, after lower-casing and punctuation normalization. Their similarity is

$$\mathrm{sim}(\hat{e}, e) = \frac{|T(\hat{e}) \cap T(e)|}{\min(|T(\hat{e})|, |T(e)|)},$$

and the answer is correct if $\mathrm{sim}(\hat{e}, e) \geq \tau_{\text{ent}}$, with similarity threshold $\tau_{\text{ent}} = 1.0$, by default. A sensitivity analysis for τ is given in Section 7.2. This admits variations of the same entity (e.g., "Yosemite Park" vs. "Yosemite National Park"), addressing the well-known *strictness problem* of exact match and F1 in open-ended QA [9, 32]. For questions with multiple gold answers, an answer is correct if the criterion above holds for any of them, following [21].

Numeric answers. A generated answer $\hat{y}$ is correct if $|\hat{y} - y|/|y| \leq \tau_{\text{num}}$, with relative error tolerance $\tau_{\text{num}} = 0.10$ following [41]. Thresholding bounds the influence of outliers (e.g., unit-conversion errors that yield orders-of-magnitude deviations and would otherwise dominate a mean relative error), and follows the relaxed-accuracy convention used for numeric answers [23].

Answer generation and parsing. After the agent issues a FINISH action, the raw SQL result rows are converted into a structured response for evaluation. On GS-QA, LLM calls generate a natural-language answer from the result rows and parse it into a standardized JSON object with normalized units; each answer is compared against all ground-truth answers, reporting only the best match [27]. On MapQA, the agent's answer is evaluated directly against the ground-truth string.

7 Experimental Results

We organize our analysis into three parts: overall performance across both benchmarks and the relationship between spatial complexity and accuracy gains (Section 7.1), robustness of accuracy across tolerance thresholds (Section 7.2), and cost efficiency (Section 7.3).

7.1 Overall Performance

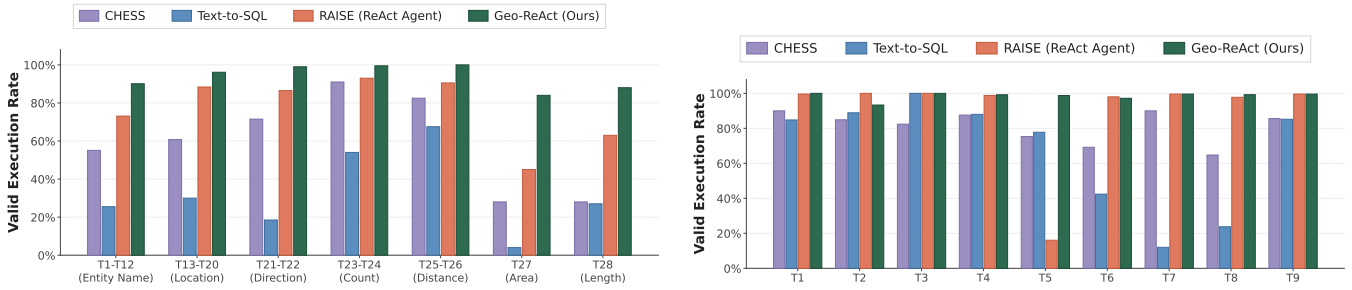
Table 2 summarizes the main results on GS-QA and MapQA. For each technique, we report the *accuracy @τ*, grouped by textual and numeric answer type. The table also reports other performance metrics including the percentage of valid queries (VER), number of LLM calls, number of tokens, and overall latency.

Geo-ReAct achieves the highest entity and numeric accuracy on both benchmarks. On GS-QA, it outperforms the strongest baseline (RAISE) by a substantial margin on both entity and numeric accuracy, with the numeric gap being considerably larger than the entity gap. On MapQA, the entity improvement over RAISE is more modest, while the numeric improvement is the largest. These results indicate that Geo-ReAct's advantage is most pronounced when questions require spatial metric computation rather than entity retrieval alone.

Geo-ReAct also achieves the highest valid execution rate (VER) (Figure 2) on both benchmarks, indicating that its concept-aligned tools help the agent produce executable queries more reliably than baselines that must compose spatial predicates without domain-specific guidance. The single-pass Text-to-SQL baseline exhibits the lowest VER, particularly on templates requiring compositional spatial predicates, where it frequently generates syntactically invalid or semantically incorrect PostGIS queries without the opportunity for iterative correction.

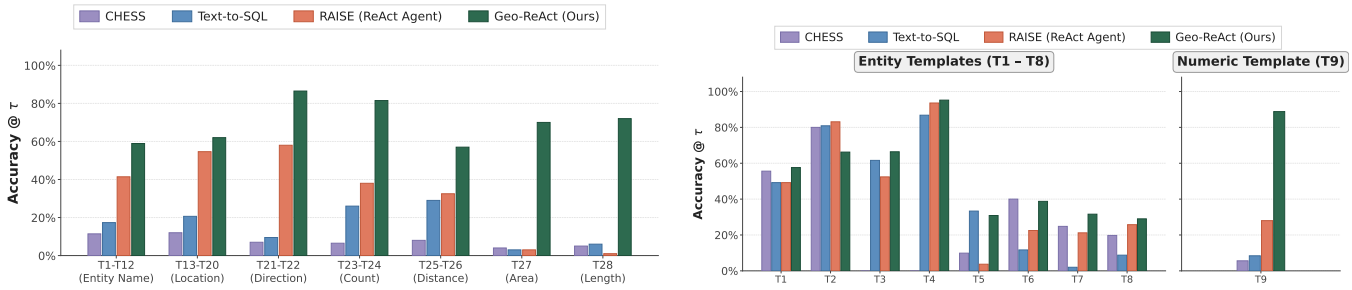
The per-template accuracy breakdown (Figure 3) reveals that the magnitude of Geo-ReAct's improvement varies systematically with the spatial complexity of the question type. We group GS-QA templates by their dominant spatial predicate and summarize the patterns below.

Range and nearest-neighbor queries. On templates that require a single spatial predicate, either a radius constraint or a nearest-neighbor selection (T1, T2, T13, T14, T17), the gap between Geo-ReAct and RAISE is narrower than on more complex templates.



(a) GS-QA (T1–T28). (b) MapQA (T1–T9).

Figure 2: Per-template valid execution rate (VER) - fraction of questions for which the system returns a non-empty answer without execution errors (%) on GS-QA and MapQA. GS-QA results are averaged over templates grouped by output type [27].



(a) GS-QA (2,800 questions, 28 templates). (b) MapQA (2,206 questions, 9 templates).

Figure 3: Per-template accuracy on GS-QA and MapQA. Entity templates appear on the left panel of each subfigure; numeric templates on the right. GS-QA results are averaged over templates grouped by output type as defined in [27].

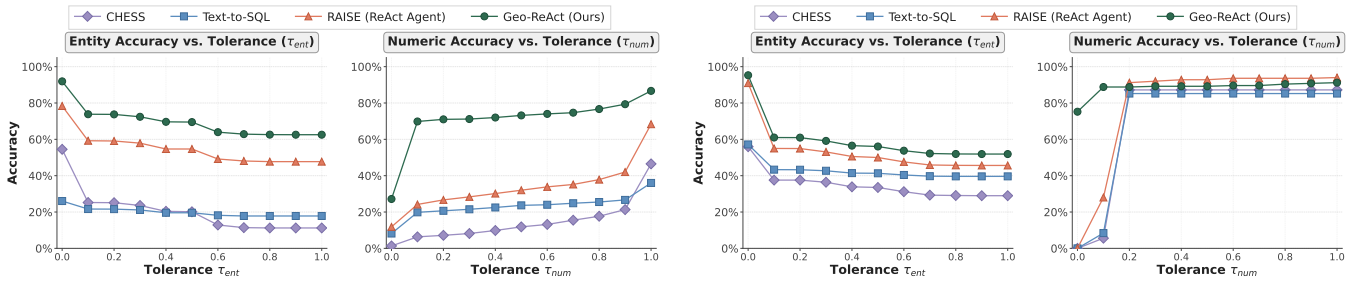
RAISE matches or slightly exceeds Geo-ReAct on a small number of these simpler templates (T2, T6, T14, T18), where general-purpose SQL exploration is sufficient to construct the required single-predicate query.

Directional reasoning. Templates combining distance with directional constraints (T3, T9, T15, T19) and templates requiring azimuth computation as output (T21, T22) show a marked separation between Geo-ReAct and all baselines (Figure 3a). Text-to-SQL achieves near-zero accuracy on the direction-filtered templates (T3, T9, T15, T19) and remains low on azimuth-output templates (T21, T22), as generating correct directional cone predicates and azimuth calculations in a single pass proves infeasible. RAISE performs better through iterative refinement but still falls well short, as it must rediscover the correct geometric formulations and coordinate conventions at each invocation. Geo-ReAct's tools encapsulate these computations, eliminating this rediscovery overhead.

Dual-anchor constraints. Templates requiring two reference locations, a source and a directional target (T4, T10, T16, T20), exhibit a consistent accuracy gradient: Geo-ReAct outperforms RAISE, which in turn outperforms Text-to-SQL across all four dual-anchor templates. Absolute accuracy decreases for all methods as the spatial constraint tightens from range-based selection (T4, T16) to nearest-neighbor selection (T10, T20).

Region-intersect queries. The largest single accuracy gain occurs on the region-intersect template T11, where Geo-ReAct outperforms RAISE by the widest margin observed across all entity templates. T12 follows a similar pattern, with Geo-ReAct substantially outperforming baselines that achieve near-zero accuracy. RAISE and Text-to-SQL both struggle because these templates require resolving an administrative boundary to its polygon geometry and performing a spatial intersection, an operation that demands both correct schema resolution and correct geometry-type selection. Geo-ReAct's GetRegionGeometry tool retrieves the polygon boundary, and GetPOIFilter ensures the appropriate geometry type is selected for the downstream computation. Without these tools, baselines either fail to locate the boundary polygon or apply the intersection to point geometries, producing empty or incorrect results.

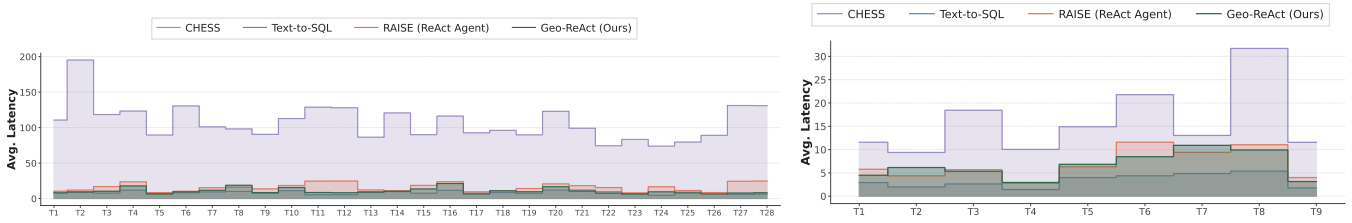
Analysis on MapQA. On MapQA (Figure 3b), Geo-ReAct's gains concentrate on the spatially harder template types. The comparison template (T5) and the directional nearest-neighbor template (T6) show the largest entity improvements, consistent with the GS-QA finding that multi-constraint spatial predicates benefit most from dedicated tools. On simpler templates such as range queries (T2) and attribute lookup (T4), baselines remain competitive, with RAISE exceeding Geo-ReAct on T2.



(a) GS-QA (2,800 questions, 28 templates).

(b) MapQA (2,206 questions, 9 reasoning types).

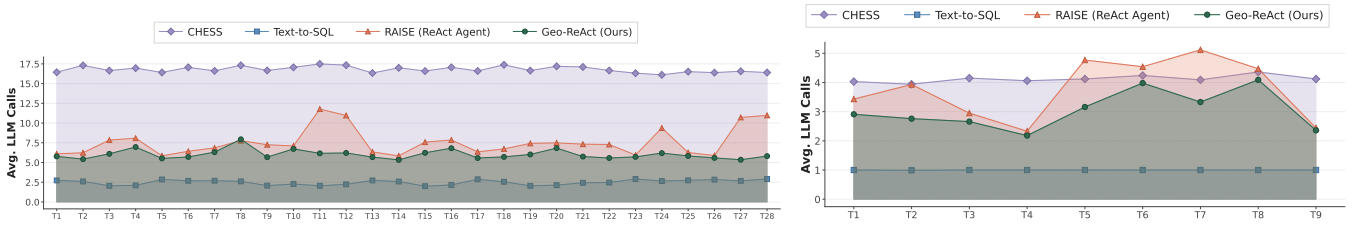
Figure 4: Entity and numeric accuracy as a function of tolerance τ on GS-QA and MapQA.



(a) GS-QA (T1–T28).

(b) MapQA (T1–T9).

Figure 5: Per-template average latency (in seconds) on GS-QA and MapQA.



(a) GS-QA (T1–T28).

(b) MapQA (T1–T9).

Figure 6: Per-template average LLM calls on GS-QA and MapQA.

7.2 Robustness Across Tolerance Thresholds

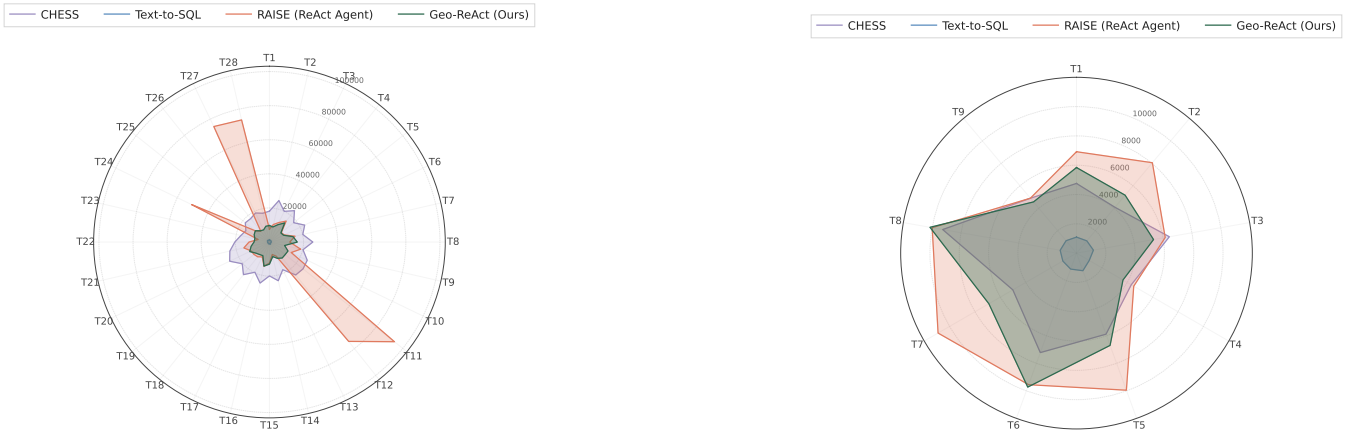
Figure 4 plots entity and numeric accuracy as a function of the matching tolerance τ on both benchmarks. On entity accuracy, Geo-ReAct maintains the highest accuracy at every tolerance level.

For numeric templates, on GS-QA (Figure 4a), Geo-ReAct maintains a wide separation from all baselines across the full tolerance range. On MapQA (Figure 4b), Text-to-SQL and CHESS achieve near-zero numeric accuracy at the primary threshold ($\tau_{num} = 0.1$), and RAISE achieves low accuracy at the same threshold, but all three baselines improve sharply at $\tau_{num} = 0.2$. This indicates that baselines produce distance estimates with systematic relative errors, consistent with coordinate-system or unit-conversion errors. Geo-ReAct, by contrast, already achieves high accuracy at the strict $\tau_{num} = 0.1$ threshold, demonstrating that its CalculateMetric tool computes distances with the precision that the dedicated PostGIS spatial functions provide, rather than relying on the LLM to construct these computations from scratch.

7.3 Cost and Efficiency

Beyond accuracy, Geo-ReAct achieves favorable cost and efficiency relative to the baselines, as shown in Table 2 and Figures 2–7.

Token consumption. Geo-ReAct uses fewer total tokens per question than RAISE on both benchmarks (Table 2), despite achieving substantially higher accuracy. The per-template token analysis (Figure 7) reveals the source of this efficiency gap: RAISE exhibits extreme token consumption spikes on region-intersect and aggregation templates in GS-QA, where the agent enters an extended exploration phase. Geo-ReAct's grounding tools resolve these mappings in a single call, avoiding the exploratory overhead. CHESS consumes a consistently high token budget due to its multi-agent retrieval pipeline, regardless of question complexity. Text-to-SQL uses the fewest tokens as a fixed single-pass pipeline, but this efficiency comes at the cost of the lowest accuracy and VER.



(a) GS-QA (T1–T28). **(b) MapQA (T1–T9).**

Figure 7: Per-template average token consumption on GS-QA and MapQA.

LLM calls. Geo-ReAct requires fewer LLM calls per question than RAISE on both benchmarks (Figure 6). CHESS requires the most calls due to its fixed four-agent architecture, which invokes multiple LLM stages regardless of question difficulty. RAISE's call count varies widely across templates, increasing on the same region-intersect and aggregation templates where its token consumption is highest, reflecting the iterative retry loops these difficult templates trigger. Geo-ReAct maintains a stable call budget across all template types.

Latency. The latency of CHESS is an order of magnitude higher than all other methods on GS-QA (Figure 5a), driven by its sequential multi-agent pipeline and retrieval stages. Geo-ReAct achieves lower latency than RAISE on both benchmarks, with Text-to-SQL being the fastest due to its single-pass design.

7.4 Impact of Spatial Tools

Geo-ReAct's accuracy advantage over RAISE, which shares the same ReAct loop but lacks spatial tools, confirms their collective value. To understand how individual tools contribute across question types, we construct a tool-template activation matrix (Figures 8a and 8b) that records, for each (template, tool) pair, the percentage of correctly answered questions in which the agent invoked that tool. The spatial tools activate selectively rather than uniformly: each tool concentrates on the templates matching its intended spatial concept. On GS-QA, GetRegionGeometry activates predominantly on region-intersect templates and on aggregation templates requiring administrative boundaries (T27, T28), while ResolveLocation drops to near-zero on those same templates. This substitution shows the agent correctly distinguishes point-based grounding from polygon-based region retrieval. Similarly, on MapQA, CalculateMetric activates near 100% on comparison (T5), directional-neighbor (T6), and distance (T9) templates but remains inactive elsewhere. ParseQuestion shows the lowest overall weighted activation (0.5% on GS-QA, 0.1% on MapQA), as it is primarily invoked on the most complex multi-constraint questions where the agent's overall accuracy is lowest, resulting in few correct predictions that include it.

Critically, the templates with the highest spatial tool activation are precisely these where Geo-ReAct achieves the largest accuracy improvements over baselines. On GS-QA, the region-intersect and aggregation templates where GetRegionGeometry and GetPOIFilter concentrate their activation are the templates where RAISE performance drops. On MapQA, CalculateMetric's activation on T5 and T9 coincides with the two largest accuracy gaps over RAISE. Conversely, on templates where spatial tools rarely activate, such as attribute lookup (T4 on MapQA, where GenerateSQL handles the query directly), Geo-ReAct and RAISE achieve similar accuracy. This co-occurrence of high spatial tool activation with large accuracy gains, and low activation with small gains, strongly suggests that the spatial tools are the mechanism behind Geo-ReAct's improvements. Meanwhile, GenerateSQL remains the most frequently activated tool overall (97% on GS-QA, 77% on MapQA), confirming that the spatial tools augment rather than replace SQL generation, and the low activation of RefineSQL (14% on GS-QA, 6% on MapQA) indicates that upstream spatial grounding reduces the need for iterative query repair. Ablation experiment results are shown in Table 5.

8 Conclusion

We presented Geo-ReAct, an agentic framework that equips an LLM with spatially principled tools derived from Kuhn's core concepts of spatial information. By aligning each tool with a well-defined spatial concept, namely Location, Neighborhood, Object, and Field, the framework encapsulates the geometric expertise that LLMs lack while preserving their natural strength in language understanding and compositional planning. Evaluation on two geospatial QA benchmarks confirms that this concept-grounded design substantially outperforms baselines across entity retrieval, numeric computation, and query execution reliability, with the most significant improvements on questions involving directional reasoning, spatial aggregation, and multi-constraint predicates. The framework achieves these gains while reducing token consumption and latency, demonstrating that principled tool abstraction benefits both accuracy and efficiency.

References

[1] Temitope Akinboyewa, Zhenlong Li, Huan Ning, and M. Naser Lessani. 2025. GIS Copilot: Towards an Autonomous GIS Agent for Spatial Analysis. *International Journal of Digital Earth* 18, 1 (2025), 2497489. doi:10.1080/17538947.2025.2497489

[2] Riyang Bao, Cheng Yang, Dazhou Yu, Zhexiang Tang, Gengchen Mai, and Liang Zhao. 2026. Spatial-Agent: Agentic Geo-spatial Reasoning with Scientific Core Concepts. *arXiv preprint arXiv:2601.16965* (2026). https://arxiv.org/abs/2601.16965 Preprint.

[3] Gheorghe Comanici, Eric Bieber, Mike Schaekermann, Ice Pasupat, Noveen Sachdeva, Inderjit Dhillon, Marcel Blistein, Ori Ram, Dan Zhang, Evan Rosen, et al. 2025. Gemini 2.5: Pushing the frontier with advanced reasoning, multimodality, long context, and next generation agentic capabilities. *arXiv preprint arXiv:2507.06261* (2025).

[4] 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. 2025. MapEval: A Map-Based Evaluation of Geo-Spatial Reasoning in Foundation Models. In *Proceedings of the 42nd International Conference on Machine Learning (ICML) (Proceedings of Machine Learning Research, Vol. 267)*. PMLR, 13774–13813.

[5] Alishiba Dsouza, Nicolas Tempelmeier, Ran Yu, Simon Gottschalk, and Elena Demidova. 2021. WorldKG: A World-Scale Geographic Knowledge Graph. In *Proceedings of the 30th ACM International Conference on Information and Knowledge Management (CIKM)*. ACM. doi:10.1145/3459637.3482023

[6] Fernando Granado, Roberto Lotufo, and Jayr Pereira. 2025. RAISE: Reasoning Agent for Interactive SQL Exploration. *arXiv preprint arXiv:2506.01273* (2025).

[7] Mordechai Haklay and Patrick Weber. 2008. Openstreetmap: User-generated street maps. *IEEE Pervasive computing* 7, 4 (2008), 12–18.

[8] Pengyue Jia, Yingyi Zhang, Xiangyu Zhao, and Sharon Li. 2026. GeoArena: Evaluating Open-World Geographic Reasoning in Large Vision-Language Models. arXiv:2509.04334 [cs.CV] https://arxiv.org/abs/2509.04334

[9] Ehsan Kamalloo, Nouha Dziri, Charles Clarke, and Davood Rafiei. 2023. Evaluating open-domain question answering in the era of large language models. In *Proceedings of the 61st annual meeting of the Association for Computational Linguistics (volume 1: long papers)*. 5591–5606.

[10] Nikolaos Karalis, Georgios M. Mandilaras, and Manolis Koubarakis. 2019. Extending the YAGO2 Knowledge Graph with Precise Geospatial Knowledge. In *The Semantic Web – ISWC 2019 (Lecture Notes in Computer Science, Vol. 11779)*. Springer, 181–197. doi:10.1007/978-3-030-30796-7_12

[11] Kiran Kate, Tejaswini Pedapati, Kinjal Basu, Yara Rizk, Vijil Chenthamarakshan, Subhajit Chaudhury, Mayank Agarwal, and Ibrahim Abdelaziz. 2025. LongFuncEval: Measuring the effectiveness of long context models for function calling. *arXiv preprint arXiv:2505.10570* (2025).

[12] Ali Khosravi Kazazi, Zhenlong Li, M Naser Lessani, and Guido Cervone. 2025. From Questions to Queries: An AI-powered Multi-Agent Framework for Spatial Text-to-SQL. *arXiv preprint arXiv:2510.21045* (2025).

[13] 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 *The Semantic Web – ISWC 2023 (Lecture Notes in Computer Science, Vol. 14266)*. Springer, 266–284. doi:10.1007/978-3-031-47243-5_15

[14] 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 (2024), 104203.

[15] Werner Kuhn. 2012. Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science* 26, 12 (2012), 2267–2276. doi:10.1080/13658816.2012.722637

[16] Werner Kuhn and Andrea Ballatore. 2015. Designing a Language for Spatial Computing. In *AGILE 2015: Geographic Information Science as an Enabler of Smarter Cities and Communities*. 309–326. doi:10.1007/978-3-319-16787-9_18 Best Paper Award.

[17] Fei Lei, Yibo Yang, Wenxiu Sun, and Dahua Lin. 2025. Mcpverse: An expansive, real-world benchmark for agentic tool use. *arXiv preprint arXiv:2508.16260* (2025).

[18] Fangjun Li, David C Hogg, and Anthony G Cohn. 2024. Advancing spatial reasoning in large language models: An in-depth evaluation and enhancement using the stepgame benchmark. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18500–18507.

[19] Zekun Li, Malcolm Grossman, Ehsan Qasemi, Mihir Kulkarni, Muhao Chen, and Yao-Yi Chiang. 2025. Benchmarking Geospatial Question Answering with MapQA. In *Proceedings of the 33rd ACM International Conference on Advances in Geographic Information Systems (SIGSPATIAL)*. ACM. doi:10.1145/3748636.3764174

[20] Zhenlong Li and Huan Ning. 2023. Autonomous GIS: the next-generation AI-powered GIS. *International Journal of Digital Earth* 16, 2 (2023), 4668–4686. doi:10.1080/17538947.2023.2278895

[21] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. When not to trust language models: Investigating effectiveness of parametric and non-parametric memories. In *Proceedings of the 61st annual meeting of the association for computational linguistics (volume 1: Long papers)*. 9802–9822.

[22] Rohin Manvi, Samar Khanna, Gengchen Mai, Marshall Burke, David B. Lobell, and Stefano Ermon. 2024. GeoLLM: Extracting Geospatial Knowledge from Large Language Models. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2310.06213

[23] Ahmed Masry, Xuan Long Do, Jia Qing Tan, Shafiq Joty, and Enamul Hoque. 2022. Chartqa: A benchmark for question answering about charts with visual and logical reasoning. In *Findings of the association for computational linguistics: ACL 2022*. 2263–2279.

[24] Dharmen Punjani, Kuldeep Singh, Andreas Both, Manolis Koubarakis, Iosif Angelidis, Konstantina Bereta, Themis Beris, Dimitris Bilidas, Theofilos Ioannidis, Nikolaos Karalis, Christoph Lange, Despina-Athanasia Pantazi, Christos Papaloukas, and Georgios Stamoulis. 2018. Template-Based Question Answering over Linked Geospatial Data. In *Proceedings of the 12th Workshop on Geographic Information Retrieval (GIR)*. ACM, 7:1–7:10. doi:10.1145/3281354.3281362

[25] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. 2024. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-world APIs. In *International Conference on Learning Representations (ICLR)*.

[26] Jonathan Roberts, Timo Lüddecke, Sowmen Das, Kai Han, and Samuel Albanie. 2023. GPT4GEO: How a language model sees the world's geography. *arXiv preprint arXiv:2306.00020*.

[27] Majid Saeedan, Muhammad Shihab Rashid, Ahmed Eldawy, and Vagelis Hristidis. 2026. GS-QA: A Benchmark for Geospatial Question Answering. *arXiv preprint arXiv:2605.22811* (2026).

[28] Simon Scheider, Enkhbold Nyamsuren, Han Kruiger, and Haiqi Xu. 2021. Geo-analytical question-answering with GIS. *International Journal of Digital Earth* 14, 1 (2021), 1–14. doi:10.1080/17538947.2020.1738568

[29] 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.

[30] Zhengxiang Shi, Qiang Zhang, and Aldo Lipani. 2022. Stepgame: A new benchmark for robust multi-hop spatial reasoning in texts. In *Proceedings of the AAAI conference on artificial intelligence*, Vol. 36. 11321–11329.

[31] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2023. Reflexion: Language Agents with Verbal Reinforcement Learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, Vol. 36. https://arxiv.org/abs/2303.11366

[32] Chenglei Si, Chen Zhao, and Jordan Boyd-Graber. 2021. What's in a name? answer equivalence for open-domain question answering. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*. 9623–9629.

[33] Michael Staniek, Raphael Schumann, Maike Züfle, and Stefan Riezler. 2024. Text-to-OverpassQL: A Natural Language Interface for Complex Geodata Querying of OpenStreetMap. *Transactions of the Association for Computational Linguistics* 12 (2024), 562–575. doi:10.1162/tacl_a_00654

[34] Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. CHESS: Contextual Harnessing for Efficient SQL Synthesis. *arXiv preprint arXiv:2405.16755* (2024). https://arxiv.org/abs/2405.16755

[35] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. 2025. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. In *Proceedings of the 31st International Conference on Computational Linguistics (COLING)*. 540–557.

[36] Zhongyuan Wang, Richong Zhang, Zhijie Nie, and Jaein Kim. 2024. Tool-assisted agent on sql inspection and refinement in real-world scenarios. *arXiv preprint arXiv:2408.16991* (2024).

[37] Huayi Wu, Zhangxiao Shen, Shuyang Hou, Haoyue Jiao, Ziqi Liu, Lutong Xie, Jianyuan Liang, Yaxian Qing, Xiaopu Zhang, Zhipeng Gui, et al. 2025. AutoGEEval++: A multi-level and multi-geospatial-modality automated evaluation framework for large language models in geospatial code generation on Google Earth Engine. *Big Earth Data* 9, 4 (2025), 747–796.

[38] Sonila Xhafa and Albana Kosovrasti. 2015. Geographic information systems (GIS) in urban planning. *European Journal of Interdisciplinary Studies* 1, 1 (2015), 74–81.

[39] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. ReAct: Synergizing Reasoning and Acting in Language Models. In *International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/2210.03629

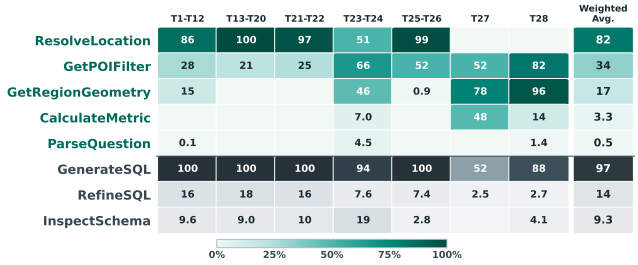
[40] Dazhou Yu, Riyang Bao, Ruiyu Ning, Jinghong Peng, Gengchen Mai, and Liang Zhao. 2025. Spatial-rag: Spatial retrieval augmented generation for real-world geospatial reasoning questions. *arXiv preprint arXiv:2502.18470* (2025).

[41] Yipeng Zhang, Chen Wang, Yuzhe Zhang, and Jacky Jiang. 2025. Text to Query Plans for Question Answering on Large Tables. *arXiv preprint arXiv:2508.18758* (2025).

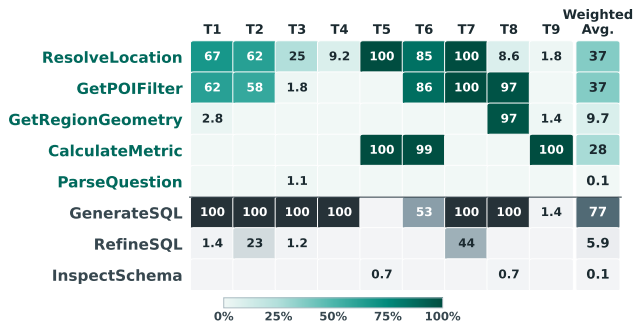
[42] Yifan Zhang, Cheng Wei, Zhengting He, and Wenhao Yu. 2024. GeoGPT: An assistant for understanding and processing geospatial tasks. *International Journal of Applied Earth Observation and Geoinformation* 131 (2024), 103976. doi:10.1016/j.jag.2024.103976

Table 3: Representative questions from GS-QA [27] (28 templates, 2,800 pairs; United States OSM) and MapQA [19] (9 types, 2,206 pairs; Southern California OSM), grouped by output type and spatial predicate. †Adds a non-spatial attribute filter (e.g., cuisine, museum sub-type). ‡Requires multi-hop reasoning over an external knowledge source.

GS-QA		MapQA
Output: Entity name GS-QA T1–T12		
Range T1 *Can you suggest a fast food restaurant within 150 km of Charlotte's Quest Nature Center, Manchester, MD?*	*Range + attribute filter†* T2 *Would you find me a history museum within 170 km from SEA LIFE Minnesota Aquarium, Bloomington, MN?*	*Range* M2 *What are the bars within 50 m of South Park Brewing?*
Nearest neighbor (NN) T5 *Can you recommend the nearest attraction from Speert Hall, Wayne, NJ?*	*NN + attribute filter†* T6 *What's the nearest local museum from Comfort Inn, Oskaloosa, IA?*	*NN (adjacency)* M1 *What restaurant is adjacent to Luther Burbank Savings?*
Range + direction T3 *Which university can I find northeast of Legoland Discovery Center Chicago, Schaumburg, IL, within 130 km?*	*NN + direction* T9 *Which museum is nearest to the south of 777 Zip/the Outlook, Davis, OK?*	*NN + direction* M6 *What is the nearest snack cart west of Colours Wheelchairs?*
Range + towards (dual anchor) T4 *Which fast food is available approaching Orange County Surgical Specialists, CA within 20 km from Sugarplum Zoo, Temecula, CA?*	*NN + towards (dual anchor)* T10 *What is the closest restaurant from Speert Green, Wayne, NJ towards Dwarfskill Preserve, Milford, PA?*	*Comparison: closer of two* M5 *Which is closer to ChargePoint, Chicken Maison or Ospi?*
Region intersect, max area T11 *Can you tell me the largest lake in Douglas County, Nebraska?*	*Region intersect, max length* T12 *Can you name the longest canal in Logan County, Colorado?*	*NN + road junction* M8 *Which driving school is nearest to the junction of 10th Street and 10th Street West?*
Output: Location (coordinates / address) GS-QA T13–T20		
Range T13 *Where can I find a university that's 180 km from Silver Point Interpretive Overlook, OR?*	*Range + attribute filter†* T14 *Could you locate a Hawaiian restaurant within 140 km from Ukrainian Cultural Garden, Cleveland, OH?*	*Range, amenity attribute* M3 *What amenities are within a 100 m radius of Yorba Rose?*
Nearest neighbor T17 *Where's the nearest restaurant from New Mexico Slaughter House, Albuquerque, NM?*	*NN + attribute filter†* T18 *Where is the nearest fusion restaurant from Pulmonary Medicine Institute, Omaha, NE?*	*Nearest neighbor* M7 *What is the nearest clinic to East Wing?*
Range + direction T15 *Where can I find an attraction towards the west of Whitney Avenue Community Garden, Lexington, KY within 160 km?*	*NN + direction* T19 *Where is the nearest hospital towards the south of Falls Ridge Preserve, Elliston, VA?*	—
Range + towards (dual anchor) T16 *Where can I find a café in the direction of Wilkinson Village Heritage Museum, MI within 80 km from King's Grove, IL?*	*NN + towards (dual anchor)* T20 *Where is the closest fast food from Allen Park Civic Arena, MI towards Detroit Artists Market, Detroit, MI?*	—
Output: Direction (azimuth) GS-QA T21–T22		
Range, direction as output T21 *In what direction can I find a zoo from South Park Rail Como Project, Como, CO within 140 km?*	*NN, direction as output* T22 *What's the course towards the closest beach resort from Detroit Lakes Sports Arena, MN?*	—
Output: Numeric / aggregate GS-QA T23–T28		
Range → count T23 *Within 130 km from Marvin Popcorn Sutton's Homeplace, Waynesville, NC, how many nature reserves are available?*	*Region intersect → count* T24 *What is the total number of restaurants in Benzonia Township, Michigan?*	—
Range → distance T25 *What's the distance to a park from Dunnellon High School Football Stadium, FL within 30 km?*	*NN → distance* T26 *What's the proximity of the nearest hospital relative to Algonquin Woods, IL?*	*Point-to-point distance* M9 *How far is 119th Street & Laflin from 100 Forest Place?*
Region intersect → total area T27 *What is the overall area of all reservoirs in Lower Turkeyfoot Township, Pennsylvania?*	*Region intersect → total length* T28 *What is the overall length of all streams in Walton County, Georgia?*	—
Output: Attribute retrieval & multi-hop GS-QA T7–T8		
NN → out-of-schema attribute‡ T7 *How many spectators can the nearest stadium from Dells Mill and Museum, Augusta, WI hold?*	*Externally described anchor → NN‡* T8 *What's the closest restaurant to the hospital with a pediatric emergency department in Bronx, NY?*	*Attribute lookup at location* M4 *What amenity is available at Port Police?*



(a) GS-QA (28 templates in 7 groups, 2,800 questions)



(b) MapQA (9 templates, 2,206 questions)

Figure 8: Tool activation rate among correct predictions. Each cell shows the percentage of correctly answered questions (at τ_{ent}=1.0, τ_{num}=0.1) that invoked the given tool. Tools are grouped into spatial (concept-aligned) and general (SQL substrate). The rightmost column reports the weighted average across templates *(weighted by the number of correct predictions per template)*.

Table 5: Spatial-tool ablation on GS-QA, using a 10% stratified sample per template. Tools are removed cumulatively, least-used first: each row drops the listed tool *in addition to* all tools above it. Δ is the accuracy drop versus the full toolset in percentage points. Degradation concentrates in *numeric* templates, while *entity* accuracy stays roughly flat until the geocoding tool *(ResolveLocation)* is removed.

	Entity		Numeric	
Tools removed (cumulative)	Acc. @ τ_{ent}	Δ	Acc. @ τ_{num}	Δ
Full toolset	60.00	0.00	70.00	0.00
− ParseQuestion	59.30	0.70	67.35	2.65
· · · − CalculateMetric	61.70	−1.70	62.05	7.95
· · · − GetRegionGeometry	62.27	−2.27	32.90	37.10
· · · − GetPOIFilter	56.82	3.18	30.25	39.75
· · · − ResolveLocation	50.45	9.55	27.60	42.40

† The transient entity accuracy gain upon removing CalculateMetric and GetRegionGeometry reflects their numeric specialization: their presence introduces minor routing overhead on entity queries. The concurrent sharp drop in numeric accuracy (−7.95 and −37.10 pp) confirms their essential contribution to numeric reasoning.

Table 4: Per-template results on GS-QA. Bold: best system per metric. CH = CHESS, T2S = Text-to-SQL, RA = RAISE, GR = Geo-ReAct. We report *Parsed F1*, *Angular Error*, and *Relative Error* following [27].

		Accuracy @ τ				VER				Aux. Metrics			
	Type	CH	T2S	RA	GR	CH	T2S	RA	GR	CH	T2S	RA	GR
Entity (T1–T12)													*Parsed F1* ↑
T1	Ent.	0.20	0.54	0.68	**0.79**	0.81	0.63	0.87	0.87	0.29	0.88	0.88	**0.97**
T2		0.05	0.48	**0.86**	0.85	0.44	0.48	0.94	0.97	0.23	**1.00**	0.94	0.96
T3		0.12	0.02	0.57	**0.75**	0.81	0.03	0.85	0.99	0.20	0.68	0.77	**0.90**
T4		0.32	0.08	0.49	**0.80**	0.62	0.09	0.67	0.92	0.47	**1.00**	0.85	0.95
T5		0.07	0.42	0.66	**0.90**	0.82	0.59	0.96	0.99	0.09	0.75	0.72	**0.88**
T6		0.06	0.35	**0.58**	0.46	0.46	0.59	0.97	0.98	0.12	0.67	0.66	**0.68**
T7		0.08	0.00	0.32	**0.57**	0.53	0.00	0.50	0.74	0.18	0.00	0.68	**0.73**
T8		0.09	0.16	0.43	**0.45**	0.22	0.32	0.76	0.62	0.35	0.57	**0.63**	0.55
T9		0.08	0.01	0.24	**0.35**	0.68	0.05	0.91	0.98	0.13	0.26	0.35	**0.44**
T10		0.14	0.01	0.07	**0.21**	0.48	0.23	0.73	0.82	0.17	0.11	0.12	**0.31**
T11		0.09	0.00	0.06	**0.71**	0.42	0.00	0.33	0.94	0.26	0.00	0.29	**0.78**
T12		0.07	0.01	0.01	**0.23**	0.32	0.05	0.28	0.99	0.23	0.26	0.14	**0.43**
Avg					–					0.23	0.51	0.58	**0.71**
Location (T13–T20)													
T13	Loc.	0.12	0.44	0.73	**0.78**	0.76	0.59	0.94	1.00			–	
T14		0.10	0.46	**0.79**	0.77	0.50	0.51	0.98	0.98			–	
T15		0.12	0.00	0.55	**0.70**	0.75	0.00	0.87	0.98			–	
T16		0.35	0.11	0.60	**0.80**	0.58	0.11	0.76	0.94			–	
T17		0.05	0.39	0.72	**0.89**	0.73	0.63	0.99	1.00			–	
T18		0.07	0.23	**0.55**	0.44	0.36	0.45	0.96	0.97			–	
T19		0.08	0.01	0.32	**0.40**	0.63	0.03	0.93	0.99			–	
T20		0.07	0.01	0.11	**0.18**	0.55	0.08	0.64	0.83			–	
Direction (T21–T22)													*Angular Error* ↓
T21	Dir.	0.09	0.13	0.69	**0.92**	0.64	0.17	0.85	0.99	0.18	**0.00**	0.06	0.03
T22		0.05	0.06	0.47	**0.81**	0.79	0.20	0.88	0.99	0.45	0.18	0.21	**0.14**
Avg					–					0.31	0.09	0.13	**0.09**
Numeric (T23–T28)													*Relative Error* ↓
T23	Cnt.	0.07	0.50	0.59	**0.87**	0.87	0.81	0.99	1.00	0.84	0.32	0.35	**0.18**
T24		0.06	0.02	0.17	**0.76**	0.95	0.27	0.87	0.99	0.91	0.93	0.72	**0.19**
T25	Dist.	0.08	0.07	0.10	**0.25**	0.77	0.54	0.87	1.00	0.36	0.24	0.16	**0.05**
T26		0.08	0.51	0.55	**0.89**	0.88	0.81	0.94	1.00	0.80	0.29	0.27	**0.09**
T27	Area	0.04	0.03	0.03	**0.70**	0.28	0.04	0.45	0.84	0.86	0.25	0.85	**0.16**
T28	Len.	0.05	0.06	0.01	**0.72**	0.28	0.27	0.63	0.88	0.89	0.77	0.93	**0.21**
Avg					–					0.78	0.47	0.55	**0.15**