Street Art & Urban Creativity | Vol. 11, No. 7, 2025 | ISSN 2183-9956
Street Art & Urban Creativity. Scientific Journal
https://doi.org/10.62161/sauc.v11.6023

SLMs FOR NATURAL LANGUAGE DATABASE INTERACTION WITH THE SENSE CITIVERSE OF CARTAGENA CITY

KEYWORDS

SLMs
text-to-sql
Trino
Dahahub
KServe
KubeFlow
Sense Citiverse
Smart City
Data Spaces

ABSTRACT

European data spaces host rich catalogues yet rarely turn data into timely answers. We present a self-contained Natural Language data assistant for Cartagena's SENSE Citiverse. It orchestrates three SLMs (Entities, SQL, and Domain Expert) on a DataHub, Trino and KServe/Kubeflow stack. Two SLM stacks (Compact, Expanded) were evaluated on six air-quality and port-operations tasks against Gemini 2.5-Pro-LLM. Entity extraction achieved perfect table-discovery. SQL solved 4/6 (Compact) and 5/6 (Expanded); Gemini solved 6/6. The SLM design preserves data-sovereignty and reduces cost. This work demonstrates a new way to interact in the Citiverse based on Natural Language.

RECEIVED: 05 / 09 / 2025
ACCEPTED: 07 / 10 / 2025

1. Introduction

European Union (EU) data spaces promise safe, sovereign sharing of data across public agencies and private actors, yet their primary interface to humans is still the dataset catalogue. Administrations publish datasets as comma-separated values or JSON and then catalogue them for discovery. Non-technical staff can browse those catalogues but often cannot turn rows and columns into answers. The result is paradoxical: data is available, but insight is scarce. Recent work on dataspaces describes this gap clearly and argues for human-centered access to machine-oriented assets (Inglés-Romero et al., 2025). While Large Language Models (LLMs) would offer a Natural Language (NL) interface, general-purpose models impose cost, privacy, and sustainability burdens which also invite unnecessary dependence on external APIs. For city data, healthcare records, or industrial telemetry, that dependence conflicts with data sovereignty requirements and complicates compliance.

Cartagena is one of the pilot cities in the SENSE initiative (SENSE Project, 2025c), which develops "Citiverses," as interconnected digital spaces where European cities plan, simulate, and co-create public services. In this municipal setting, air-quality is an early, specific use case where Cartagena has made a large investment to develop its sustainability plans and Low Emission Zone. Cartagena aggregates environmental telemetry such as hourly air pollutants (O_3, NO_2, SO_2 and CO). Then, technicians need to answer simple but time-sensitive questions: Do we have O_3 values above a threshold on a given date? Are CO values stable over a night shift?

A conversational text-to-sql assistant would address that gap. It lets staff use natural language, then discovers the correct dataset, composes the SQL with the right quoting for spaces and diacritics, validates it, executes it, and then explains the result.

To address this demand while avoiding inherent LLM pitfalls, this paper proposes a self-contained data assistant powered by Small Language Models (SLMs), smaller LLMs that can fit on consumer devices. The data assistant accepts natural language questions, discovers relevant datasets, composes and executes SQL, and then explains the results. Three specialized SLMs are leveraged: an Entities SLM for term extraction, a SQL SLM for query generation, and a Domain Expert SLM for NL explanations powered by an open-source infrastructure end-to-end. DataHub provides table information and metadata for discovery and context, Trino serves as a federated SQL engine across files and databases, and KServe runs the SLMs on a Kubernetes cluster, providing secure, observable, and autoscaled model endpoints. Each component is well established, actively maintained, and compatible with on-premise deployment. Data never leaves the controlled environment.

We have favoured SLMs over monolithic LLMs for four reasons: First, SLMs meet latency constraints for interactive tools. They respond in hundreds of milliseconds to a few seconds on a single GPU or even a strong CPU. Second, SLMs reduce operating costs and energy use, which matters when the system must serve many users or run continuously (Belcak et al., 2025). Third, specialized models are easier to debug because each one owns a narrow contract: extract entities, write SQL, or explain results. Fourth, the modular design supports graceful degradation: A single generalist model can play multiple roles on a small machine, while a larger cluster can scale horizontally and attach a different expert to each role. In Cartagena's case, lower computational demand translates directly into the ability to self-host the models alongside city data.

The usage of Open Source components allows for no external dependencies and allows for a standalone deployment. DataHub offers a graph-based metadata service with GraphQL and REST endpoints and flexible ingestion pipelines. It indexes datasets, schemas, and lineage and supports near-real-time updates for search and discovery. These features supply the context that text-to-sql systems need for schema linking and example retrieval. Trino exposes a unified SQL interface to differently formatted files, relational databases, and object stores through a rich connector ecosystem. It also can validate queries with EXPLAIN which supports the assistant's capability for query debugging. KServe, from the KubeFlow ecosystem, allows for the SLMs to be deployed as Kubernetes Custom Resource Definitions. It abstracts model server plumbing, traffic management, and autoscaling, and it supports multi-model density through ModelMesh.

Together, these tools make the assistant portable and independent. Why insist on self-containment? Sovereignty and ecology demand it. European projects treat data sovereignty as a design constraint for dataspaces. They require that processing occurs under local control, with complete authorization and traceability (Inglés-Romero et al., 2025). A self-hosted assistant respects that constraint because both the models and the data remain in-house. It also cuts energy consumption by avoiding heavy remote inference.

The proposed approach aims for an assistant that non-experts can use without calling an engineer or a domain expert. Users write plain sentences in natural language (NL), and the system outputs the results from a SQL Query as well as a NL answer to the prompt using the SQL output. This loop mirrors agentic designs that forgo one-shot magic for reliable decomposition and iterative refinement. The value proposition is clear for municipalities and enterprises. The assistant lowers the barrier to data use, preserves privacy, and avoids vendor lock-in. It can run on standard servers and leverages open software. Its modular parts are replaceable and independently upgradable depending on the specifics of a particular use case, which can matter more than marginal gains from the largest models.

1.1. Objectives

This paper proposes and details a modular, standalone database assistant for the SENSE Citiverse of Cartagena that emphasizes European dataspaces demands through SLMs and open-source tools usage. The assistant operates through a multi-step process where it:

(1) extracts entities from natural language; (2) uses DataHub to ground those entities to concrete tables and columns; (3) prompts a SQL specialist to write and repair queries; (4) validates and executes on Trino; and returns a domain-aware explanation.

The architecture, based on DataHub, Trino, KubeFlow and KServe, has been designed considering the SENSE and EU Local Digital Twin Toolbox open source ecosystems, to guarantee that these solutions can be adopted into the existing Open Platforms widely adopted by the Smart Cities ecosystem.

The rest of the paper is organized as follows: Section 2 takes on the background and state of the art (SoTA); Section 3 describes the architecture and pipeline; Section 4 includes a practical evaluation of the system with a specific configuration for the use case of Cartagena air quality readings and Section 5 concludes the paper and addresses future research.

2. Background

2.1. Text-to-sql: Connecting humans to machines

Recent surveys chronicle three waves in text-to-sql: rule-based and sequence-to-sequence systems; pre-trained language models; and, since 2023, instruction-following LLMs (Hong et al., 2024; Shi et al., 2025). LLMs shine because they support few-shot learning and robust semantic parsing. Two implementation paradigms dominate: prompt engineering via in-context learning and model fine-tuning (Hong et al., 2024; Shi et al., 2025). Methodologically, modern systems decompose the task. Pre-processing formats schemas with structured templates ("CREATE TABLE…") and sample rows; schema linking narrows the candidate tables and columns before generation (Shi et al., 2025). Inference uses Chain-of-Thought and multi-step workflows that write SQL clause by clause or divide the prompt into sub-problems. Post-processing introduces database feedback: self-debugging with error logs, execution-guided selection, and cross-model consistency checks (Hong et al., 2024; Shi et al., 2025).

Enterprise case studies reinforce this pipeline view. LinkedIn researchers proposed a multi-agent system with context retrieval, a Query-Writer agent, and a Researcher agent that fixes table and column hallucinations (Chen et al., 2025). Despite progress, gaps remain. Even with advanced prompts and agents, execution accuracy can be low across SoTA text-to-sql benchmarks like Spider 2.0 (Shi et al., 2025) with issues such as long-context schemas exceeding token budgets, lowering accuracy but also raising latency and cost.

2.2. DataHub: Data platform and data governance

DataHub is an Open Source metadata platform that unifies discovery, governance, and observability for data and AI assets. It exposes several APIs, including a public GraphQL interface and offers OpenAPI endpoints for common use cases. Its Metadata Service, also called GMS, maintains the graph of entities and relationships and serves queries and mutations from clients. The project emphasizes a "living metadata model" that evolves with tools, dashboards, datasets, models, and training runs (DataHub Project, 2015a; DataHub Project, 2015b; DataHub Project, 2024; DataHub Project, 2025).

A flexible ingestion architecture supports push and pull modes and can stream changes for near-real-time indexing. The quickstart deployment illustrates the platform's moving parts, including Kafka, Elasticsearch or OpenSearch, and the web frontend. The default profile runs locally, which suits development and small pilots. Production deployments use the same components at larger scale and often adopt Helm charts for orchestration (DataHub Project, 2015b).

As shown in other works (Chen et al., 2025), for a text-to-sql assistant, DataHub may play two roles. First, it catalogues datasets with rich attributes, tags, owners, and lineage so that entity extraction can resolve to specific tables and columns. Second, it provides the schemas and keys that an LLM needs to generate correct joins and filters.

2.3. Trino: Distributed SQL engine

Trino is a distributed SQL query engine designed for interactive analytics over data stored in heterogeneous systems. It coordinates queries across many workers, pushes computation to data through connectors, and returns results with low latency. Trino speaks ANSI SQL, integrates with business intelligence tools, and operates at scales ranging from gigabytes to exabytes (Trino, 2025a).

The connector framework is central to Trino's value. Connectors implement a metadata interface that lists schemas, tables, and columns and a split interface that streams data to workers. Official connectors cover common systems like Hive, Iceberg, MySQL, PostgreSQL, Kafka, and Elasticsearch, and organizations can develop custom connectors to reach internal stores (Trino, 2025b).

Trino supports query validation and introspection through EXPLAIN and EXPLAIN ANALYZE. EXPLAIN prints the distributed plan and validates the statement without running it. EXPLAIN ANALYZE executes the query and reports per-operator costs such as rows, CPU, memory, and network (Trino, 2020; Trino, 2025c).

2.4. KServe: Deploying AI models on KubeFlow

KServe is a Kubernetes-native system for serving machine learning models. It began as KFServing within the Kubeflow project and later became an independent open-source initiative. KServe defines an InferenceService Custom Resource that describes how to deploy a model server, roll out new revisions, and route traffic. It abstracts autoscaling, networking, health checks, and observability and standardizes inference protocols for REST or gRPC traffic (KServe Project, 2025c; KServe Project, 2025d; KServe Project, 2025e).

KServe integrates with the broader Kubernetes ecosystem. Kubernetes manages pods, services, and horizontal scaling for containerized applications, and it supplies declarative configuration for reliable operations (Kubernetes, 2024). Furthermore, Kubeflow provides additional components for machine learning workflows on Kubernetes, including pipelines, notebooks, a training operator, and a central dashboard. KServe fits into that ecosystem as the serving layer for trained models (Kubeflow, 2025).

Two KServe features are especially relevant. First, the platform offers flexibility in its API protocols. It not only defines a framework-agnostic inference API, often called the "v2 protocol," which standardizes request and response formats, but it also directly supports the OpenAI completions API format (KServe Project, 2025b; KServe Project, 2025c). Second, ModelMesh supports high-density, multi-model serving. It loads and unloads models on demand and routes

requests with minimal overhead. These features let a single cluster host several SLM endpoints while staying responsive and resource efficient (KServe Project, 2024).

KServe's control plane manages model revisions, canary rollout, and A/B testing. Operators can configure percent-based traffic splits to compare model versions under load. Because models are Kubernetes resources, they inherit cluster-level security, monitoring, and audit capabilities. This fit is important for sovereign deployments where every request path and artifact must be traceable (KServe Project, 2025a).

2.5. Small Language Models (SLMs)

The case for SLMs rests on three claims. First, specialized tasks do not require the full generality and size of frontier models. Second, operational realities make small models faster to deploy, monitor, and update. Third, economics and ecology both favour reduced parameter counts. As proposed by Belcak et al. (2025): an SLM fits on common consumer hardware and supports agentic interactions with low latency.

Agentic design also plays to SLM strengths. In our case, when the system isolates entity extraction, schema linking, code generation, and explanation, each sub-task benefits from a model that can be fine-tuned on a narrower corpus and post-trained for strict output formats. Such models can incorporate guardrails and deterministic decoding for tables, JSON, or SQL. They also expose failures cleanly, which reduces the cost of debugging and retraining (Belcak et al., 2025). The assistant described in this paper follows that strategy. It pairs lightweight models with strong tools—DataHub for context and Trino for execution—which reduces the cognitive burden on the models and avoids unnecessary tokens.

From a governance perspective, SLMs advance accessibility and fairness. Training and operating them costs orders of magnitude less than frontier LLMs, which makes self-hosting viable for small organizations and municipalities. That democratization matters in regions with limited budgets and in projects where public money funds infrastructure. It also matters for sustainability. Smaller models also consume less energy per request, especially when quantized (Ukil et al., 2025).

2.6. Natural Language Processing on Public Administrations

Natural Language Processing (NLP) is the most widely adopted Generative AI tool in local governance. In its different guises, it already delivers value across public administrations worldwide in areas such as data governance, where NLP simplifies the management and processing of heterogeneous data (Jiang et al., 2023; Yigitcanlar et al., 2024).

Municipalities primarily deploy NLP-based solutions for information management and back-office tasks such as automating routine administrative functions and managing citizen inquiries, enhancing public services' accessibility through chatbots and virtual assistants front-ends (Yigitcanlar et al., 2024).

And, while NLP can improve transparency by facilitating access to Open Government Data, one gap still remains, existing assistants at the public administration level are often limited to retrieving documents or listing available datasets. They fall short of enabling non-technical users to further consult the data contents themselves solely through natural language, as it cannot be effectively transformed into, for example, programmatic SQL queries to retrieve the relevant data (Cortés-Cediel et al., 2023).

2.7 SENSE Citiverse and Cartagena

The SENSE project (*Strengthening Cities and Enhancing Neighbourhood Sense of Belonging*) builds a network of interconnected digital twins that mirror real cities and align with the EU Smart Communities initiative (SENSE Project, 2025c). SENSE follows a layered architecture that starts with Device Management (IoT and edge), passes through Data Management (platform and integration), adds a Data Space for sovereign exchange, enables a Digital Twin for simulation and AI, and culminates in the Citiverse for 3D/AR/VR interaction. CitiVerse is described by the SENSE project as a shared digital space where European cities use common tools and shared data,

especially digital twins, to plan, test ideas, and involve citizens in decision-making (SENSE Project, 2025a; SENSE Project, 2025b). Cartagena is one of the two pilot cities (SENSE Project, 2025a; SENSE Project, 2025b), along with Kiel, Germany, which as Cartagena, is also renowned for its port. Cartagena's pilot explicitly relies on real-time data and interoperable stacks within a sovereign dataspace and digital-twin context, therefore, a natural-language database interface could be a great leap forward.

3. Proposal

Figure 1 depicts the complete self-contained system. An Orchestrator coordinates three LLM roles: Entities, SQL, and Domain Expert; and two data services: DataHub for metadata discovery and Trino for federated SQL execution. All components run in-cluster on Kubernetes and KServe exposes each LLM role powered by an SLM as an InferenceService with autoscaling, metrics, and revision control. This deployment choice embodies the project's operational principles: modularity (each capability is a replaceable service), sovereignty (no external APIs are required), and robustness (every step is observable, validated, and recoverable).

The system embraces modularity without sacrificing simplicity. In a minimal footprint, a single instruction-tuned SLM can assume all three roles through role-specific prompts. In a scaled footprint, the Orchestrator binds each role to a distinct model: a compact extractor for entities, a code-tuned model for SQL, and a dialogue-oriented model for explanations. KServe routes traffic to these endpoints and scales them independently.

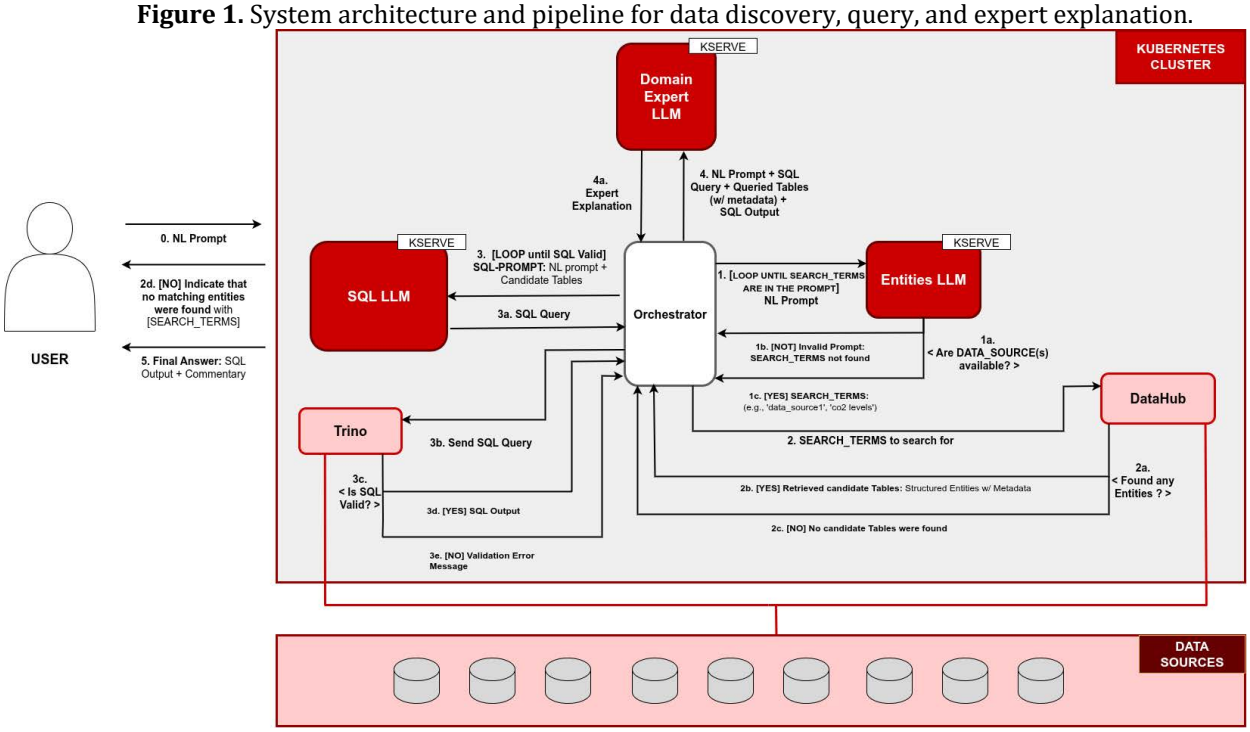
This composition allows support for different use cases, as some scenarios only need to handle a few tables with limited columns where a more general LLM suffices as all necessary context can be added in-prompt, helping it perform better, and complex queries are not the norm. However, the architecture also allows support for more complex scenarios where it is not possible to pass all context, so the Entities LLM needs to find all possible search terms from a bigger pool, the SQL LLM needs to be more precise and the Domain Expert is needed to review more complex prompts and outputs. The pipeline follows the next steps as shown in Figure 1:

1. Receive and extract entities. A user submits a natural-language prompt. The Orchestrator invokes the Entities LLM, which emits structured search terms: data sources, locations, metrics, time windows, device identifiers. Its contract is narrow by design: name things, do not infer answers.
2. Discover context. Using those terms, the Orchestrator queries DataHub to retrieve candidate tables and metadata such as column descriptions, … If no candidate tables are found, the system informs the user that their prompt is invalid.
3. Compose a focused prompt. The Orchestrator builds a structured prompt for the SQL LLM that includes the user question verbatim and the candidate tables.
    a. Generate and validate SQL. The SQL LLM returns a candidate query and Orchestrator validates it in Trino using *EXPLAIN*. If Trino reports an error—unknown column, invalid cast, or wrong function—the Orchestrator feeds the message back to the SQL LLM with a compact repair directive and revalidates.
    b. Execute and collect results. When the query compiles, the Orchestrator executes it in Trino.
4. Explain and return. The Orchestrator calls the Domain Expert SLM with the original question, the validated SQL and the result output. The model produces a concise, role-appropriate explanation that states assumptions, units, and coverage, and then the Orchestrator returns both the SQL output and the explanation to the user.

Ecological efficiency follows from specialization and sharing. The system prefers small, quantized SLMs that meet formatting and latency needs without excess capacity. ModelMesh can improve utilization by loading models only when traffic arrives. Because semantics live in DataHub and heavy computation runs in Trino, the models remain small and stateless, which lowers memory footprint and power draw while preserving capability for the target tasks. The

result is a low-power assistant that responds quickly on modest hardware across heterogeneous data sources, thanks to Trino and DataHub.

Figure 1. System architecture and pipeline for data discovery, query, and expert explanation.



Source. Own elaboration, 2025.

Finally, the architecture scales and adapts. As expressed, the Orchestrator is stateless and may run multiple replicas behind a service, so throughput scales linearly. KServe autoscaling grows or shrinks model pods with demand. Because the Orchestrator filters tables to the few that matter, token counts stay low and latency remains stable as catalogues grow. The modularity focus also means that enhancements to the architecture could be leveraged for distinct use cases. For example, a translator LLM actor could be easily added to better facilitate non-English speakers to interact with the system and a predictor model to predict future data based on current data is to be also included, but more on new modules will be discussed later on.

3.1. Front-end application: SENSE NL2SQL

The assistant's value depends on how quickly a user can turn a question into a defensible answer. The SENSE NL2SQL front-end (Figure 2) operationalizes the pipeline in Figure 1 by exposing each step—entity detection, SQL composition, execution, and explanation—in a single, comprehensible view. The interface privileges clarity over configurability so that technicians can stay in their task context and iterate rapidly. It keeps computation in-cluster and renders only the minimal information needed for decision-making, which aligns with the sovereignty and observability goals of the architecture.

3.1.1. Layout and primary flow

The front-end centers the Natural Language (NL) query box and pairs it with two explicit actions: *Suggest* and *Run Prompt*. A status chip confirms the connection to the execution engine ("Connected to Trino • Active"). To the left, *Recent Prompts* and *Quick Actions* support recall and hand-offs. The user can easily return to an earlier prompt and its output. This arrangement reduces cognitive load: users type a plain question, see system state at a glance, and re-run common analyses without navigating away. The screen then unfolds the analysis in four panels: *Generated SQL*, *Detected Entities*, *Query Status*, and *Schema Context*. Together they make the model's reasoning visible and auditable.

3.1.2. Transparent NL→SQL mapping

The *Generated SQL* panel shows the exact statement to be executed. The *Detected Entities* list reflects the Entities model's output ("air", "quality", "data", "july"), which helps users verify that the system grounded their intent against the catalogue. The *Schema Context* block summarizes the table and key columns so that a non-expert can check units and meanings before trusting a result. *Query Status* reports latency and outcome (for example, "Query executed successfully • 116 ms"), which sets expectations about interactivity and failure modes.

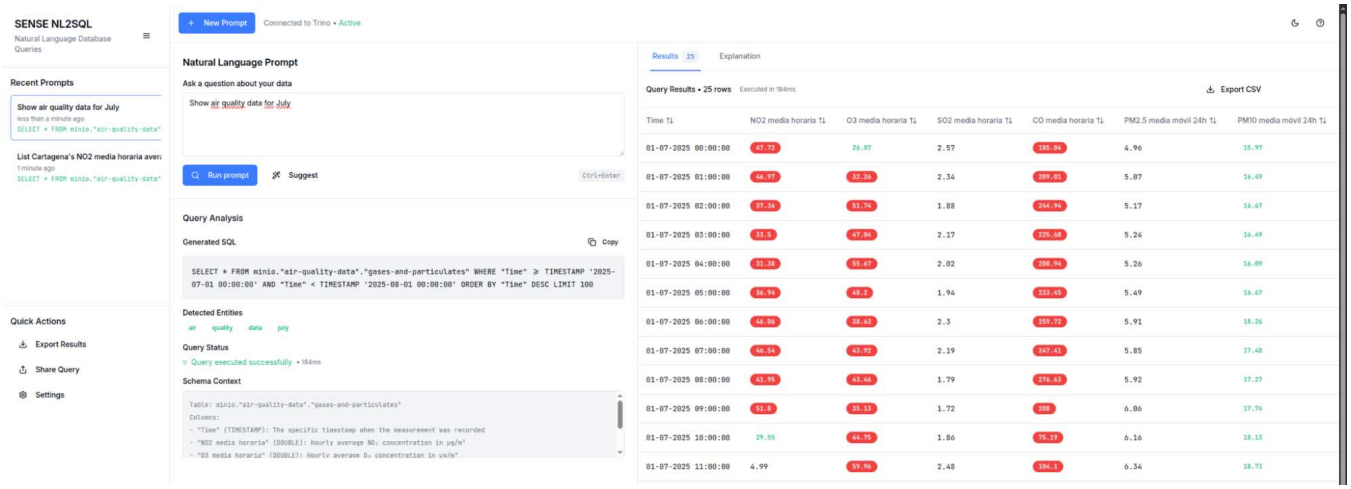
3.1.3. Results made legible

The results region presents two tabs—*Results* and *Explanation*. *Results* displays a sortable grid with the SQL output. In this case, time-stamped measurements and Spanish column names with diacritics, matching the source catalogue. This fidelity reassures users who routinely cross-check numbers against operational dashboards. *Explanation* houses the Domain Expert output for plain-language summaries, answering the user's prompt. Then, *Export CSV* allows to easily export the SQL output to a CSV file for further processing. A *Share Query* option promotes reproducibility by circulating the original question and its compiled SQL.

3.1.4. Progressive disclosure and trust

The interface practices progressive disclosure. It shows the answer first and reveals the reasoning just below. Users who need only a quick check can stop at the table and answer. Users who must audit can inspect detected entities, the SQL, and the schema context. This mirrors the agentic workflow in Figure 1 while keeping the surface area small. The result is a tool that supports both quick situational awareness and defensible reporting.

Figure 2. Complete interaction for a simple prompt. The UI displays the SQL query, the detected entities, an execution-time badge, as well as the schema and table context. The result preview is located in the left and can be toggled along the explanation window.



Source. Own elaboration, 2025

4. Evaluation

This section specifies the end-to-end evaluation of the architecture and pipeline proposed to power the assistant deployed in the context of use cases within the SENSE Citiverse of Cartagena. The protocol measures the three capabilities aligned with the pipeline explained in Figure 1: (1) entity resolution against DataHub (tables and metadata discovery), (2) SQL generation against Trino (execution and validation), and (3) domain-aware explanation (natural-language interpretation) for two SLM model configurations and a baseline SoTA LLM, Gemini 2.5 PRO. These capabilities are evaluated based on the retrieval of the correct tables, the correctness and optimization of the obtained SQL query for each task, and the quality of the answer.

4.1. Testbed

Table 1. Hardware Setup

Resource	Device
GPU	1× NVIDIA L40s-1-gpu (cloud)
GPU VRAM	45 GiB RAM
CPU	13 vCores
CPU RAM	80 GiB

Source. Own elaboration, 2025

Table 2. Models Configuration for Compact-SLM Stack

Role	HuggingFace Model ID
Entities LLM	google/gemma-2-2b-it (Google, 2025a)
SQL LLM	Qwen/Qwen2.5-Coder-3B-Instruct Qwen Team. (2025a)
Domain Expert LLM	meta-llama/Llama-3.2-3B-Instruct Meta AI. (2024a)

Source. Own elaboration, 2025

Table 3. Models Configuration for Expanded-SLM Stack

Role	HuggingFace Model id
Entities LLM	google/gemma-3-4b-it (Google, 2025b)
SQL LLM	Qwen/Qwen2.5-Coder-7B-Instruct Qwen Team. (2025b)
Domain Expert LLM	meta-llama/Llama-3.1-8B-Instruct Meta AI. (2024b)

Source. Own elaboration, 2025

Due to consumer hardware unavailability at the moment, benchmarking has been run on an NVIDIA L40 GPU. On the SLM front, we have chosen a dual model configuration approach to better highlight the tradeoffs between models. First, we have the Compact-SLM Stack (≤ 3 Billion Parameters) which prioritizes lower latency and memory footprint to meet interactive constraints and lower-end consumer hardware as well as possible edge computing support. On the other hand, the Expanded-SLM Stack (≤8 Billion Parameters) prioritizes the extra context window offered by bigger LLMs to allow for more robust SQL composition through larger and more structured prompts, which come at the cost of more pronounced hardware requirements. We compare both stacks to Gemini 2.5 Pro, used as a single generalist model that plays all roles (Entities, SQL, Domain Expert). Prompts will be the same for the Entities LLM and Domain Expert LLM, but not for the SQL LLM, as it allows larger prompts with more SQL relevant information and generic examples.

4.2. Use-case walk-through

This evaluation uses two tables with a mix of real and synthetic data: hourly air quality and cruise-ship arrivals (Tables 4–5). The air-quality table contains one record per hour and preserves Spanish column names with spaces and diacritics, while the cruise-ship arrivals have English column names. Both tables have descriptions accessible via Datahub in English. The identifiers, especially in Spanish, deliberately stress schema linking via DataHub and strict identifier quoting in SQL. Pollutants are hourly averages, although PM2.5/PM10 are 24-hour moving averages. The arrivals table encodes berth intervals as well as cruise-ship information with start and end timestamps plus capacity attributes. Together both datasets create a compact but realistic setting for entity resolution, SQL generation, and domain interpretation.

Table 4. Cartagena air-quality table: minio."air-quality-data"."gases-and-particulates"

Column	Description	SQL Data Type
"Time"	The specific timestamp (date and time) when the air quality measurement was recorded	TIMESTAMP
"NO2 media horaria"	The hourly average concentration of Nitrogen Dioxide (NO_2), a key indicator of pollution from traffic and combustion. Measured in $\mu g/m^3$.	DOUBLE
"O3 media horaria"	The hourly average concentration of ground-level Ozone (O_3), a secondary pollutant formed from other emissions in the presence of sunlight. Measured in $\mu g/m^3$	DOUBLE
"SO2 media horaria"	The hourly average concentration of Sulfur Dioxide (SO_2), primarily emitted from burning fossil fuels. Measured in $\mu g/m^3$.	DOUBLE
"CO media horaria"	The hourly average concentration of Carbon Monoxide (CO), a gas resulting from the incomplete combustion of carbon-containing fuels. Measured in mg/m^3	DOUBLE
"PM2.5 media móvil 24h"	The 24-hour moving average concentration of fine particulate matter with a diameter of 2.5 micrometers or less. These particles can penetrate deep into the lungs. Measured in $\mu g/m^3$	DOUBLE
"PM10 media móvil 24h"	The 24-hour moving average concentration of inhalable particulate matter with a diameter of 10 micrometers or less. Measured in $\mu g/m^3$.	DOUBLE

Source. Own elaboration, 2025

Table 5. Cartagena cruiseship arrivals table: minio."cruiseship_data"."arrivals_july"

Column	Description	SQL Data Type
"arrival_id"	A unique numerical identifier assigned to each individual cruise ship arrival event.	INTEGER
"cruise_ship_name"	The official, registered name of the cruise ship.	VARCHAR
"ship_size_category"	A classification of the ship based on its size or passenger capacity (e.g., Small, Medium, Large).	VARCHAR
"passenger_capacity"	The maximum number of passengers that the cruise ship is certified to carry.	INTEGER
"crew_onboard"	The total number of crew members working on the ship for that specific voyage.	INTEGER
"total_onboard"	The total number of individuals on the ship, representing the sum of passengers and crew.	INTEGER
"arrival_datetime"	The exact date and time when the ship officially docked at the port.	TIMESTAMP
"departure_datetime"	The exact date and time when the ship departed from the port.	TIMESTAMP
"berth_duration_hours"	The total time, measured in hours, that the ship remained docked at its berth.	DOUBLE

Source. Own elaboration, 2025

Six use cases progressively exercise the assistant over synthetically expanded July 2025 Cartagena data: simple to complex air-quality queries, simple to complex cruise-arrival queries, and two temporal joins. The sequence validates entity resolution, Trino-compliant SQL

generation, and domain explanations—from threshold lookups to consecutive-hour detection aligned with berth windows.

4.2.1. Use Case 1 (UC-1) — Air Quality

- NL prompt: "Do we have any hourly ozone readings bigger than or equal to 60 on 01-07-2025, and at what hours?"

UC-1 confirms basic retrieval: a single-day, thresholded ozone (O_3) query. It validates table discovery with metadata, timestamp casting, and SQL *sargable* half-open windows on "time"

4.2.2. Use Case 2 (UC-2) — Air Quality

- NL prompt: "For 02-07-2025, list the hours when SO_2 and NO_2 values are larger than the daily mean."

UC-2 raises complexity by computing same-day baselines (daily means) and filtering hours when SO_2 and NO_2 exceed those means. It tests aggregation, projection of deltas, and whether ordering by hour is applied or not.

4.2.3. Use Case 3 (UC-3) — Cruise Arrivals

- NL prompt: "Show all cruise-ship arrivals and departures scheduled on 06-07-2025, with ship name and berth duration hours."

UC-3 verifies the arrivals schedule on a single date with minimal projection and ordering to verify that the second table is also correctly retrieved and correctly processed.

4.2.4. Use Case 4 (UC-4) — Cruise Arrivals

- NL prompt: "Between 20-07-2025 at 14:30 and 22-07-2025 at 12:00, list the top 3 docked cruises by total onboard, with arrival and departure times."

UC-4 expands the window and ranks ships by total_onboard, checking that the SQL SLM uses either the provided column or a safe recomputation. These tasks confirm correct parsing of "arrival_datetime"/"departure_datetime", ordering, and limiting.

4.2.5. Use Case 5 (UC-5) - Temporal Join for Cruise Arrivals and Air Quality

- NL prompt: "For each cruise arrival on 13-07-2025, show the SO2 values per hour that fall inside its arrival and departure"

UC-5 aligns hourly air-quality measurements with each ship's berth window on 13-07-2025, returning the SO_2 value for each hour within [arrival_datetime, departure_datetime). It verifies correct interval predicates, identifier quoting with spaces/diacritics, and minimal projection/grouping per arrival without additional aggregations.

4.2.6. Use Case 6 (UC-6) — Temporal Join for Cruise Arrivals and Air Quality

- NL prompt: "Identify SO_2 values above July 2025's average for ≥ 5 consecutive hours. Then list the cruises docked during that period of time if there are any as well as the average SO_2 values during its berth and the July 2025's average."

UC-6 first detects ≥ 5 consecutive hours when SO_2 exceeds the July 2025 monthly mean, using robust run-detection over hourly timestamps. It then intersects those high-SO_2 intervals with each ship's [arrival_datetime, departure_datetime) to list any docked cruises and compute the average SO_2 during their berth, validating monthly baselines, interval logic, and precise time bounds.

4.3. Benchmark

4.3.1. Validity Checks and Labelling

This subsection defines the rubric used to score each role and the overall system in the benchmark tables that follow.

4.3.1.1. Entities LLM (table discovery)

- Optimal (✓) if the emitted search terms are sufficient to retrieve the exact table(s) required by the NL prompt (single table for UC-1/2/3/4; both tables for UC-5/6). Terms must be discriminative and aligned with schema vocabulary (including diacritics). Incorrect (x) otherwise. Retrieving the two tables when only one is required will not trigger an incorrect.

4.3.1.2. SQL LLM (query generation)

- Incorrect (x): SQL does not execute in Trino, references non-existent identifiers, misinterprets time windows, or returns wrong rows.
- Correct(✓*): Executes and returns the correct result, may not be computationally optimal.
- Optimal(✓): Executes, returns the correct result, and it is as computationally optimal as possible.
- Note: If a prompt is ambiguous and there are multiple interpretations, as long as the query is correct and returns an output compatible, it will be valid. If non-explicit ordering is included in the prompt, it is not considered suboptimal to not include ORDER BY, even if it improves usability. The same goes for unasked columns.

4.3.1.3. Domain Expert LLM (NL answer quality)

- Incorrect (x): It does not produce a correct answer to the user's NL prompt.
- Correct (✓*): Answers correctly the user's NL prompt while faithfully grounded in the SQL output and its column descriptions. Numeric data reproduced with ≤ one-decimal rounding.
- Optimal (✓): All ✓* checks plus domain context and insights for technicians such as interpretation when applicable and overall meaning.

Table 6. Notation and descriptions

Notation	Description
✓	Fully meets all correctness criteria.
✓*	Correct, but either SQL not optimal OR Domain Expert Explanation does not offer technical insights
✓**	Correct but SQL not optimal AND Domain Expert Explanation does not offer technical insights
x	Fails some correctness criteria
-	Unable to complete due to previous error

Source. Own elaboration, 2025.

4.3.2. Results

Table 7. Use Case Evaluation for Compact-SLM Stack

Use Case	Entities LLM	SQL LLM	Domain Expert LLM	Valid Output
UC-1	✓	✓	✓*	✓*
UC-2	✓	✓	x	x
UC-3	✓	✓	✓*	✓*
UC-4	✓	✓	x	x
UC-5	✓	x	-	-
UC-6	✓	x	-	-

Source. Own elaboration, 2025

Table 8. Use Case Evaluation for Expanded-SLM Stack

Use Case	Entities LLM	SQL LLM	Domain Expert LLM	Valid Output
UC-1	✓	✓*	✓*	✓**
UC-2	✓	✓*	x	x
UC-3	✓	✓	✓*	✓*
UC-4	✓	✓*	✓*	✓**
UC-5	✓	✓*	✓*	✓**
UC-6	✓	x	-	-

Source. Own elaboration, 2025.

Table 9. Use Case Evaluation for Gemini 2.5 PRO

Use Case	Entities LLM	SQL LLM	Domain Expert LLM	Valid Output
UC-1	✓	✓	✓	✓
UC-2	✓	✓	✓	✓
UC-3	✓	✓	✓	✓
UC-4	✓	✓	✓	✓
UC-5	✓	✓	✓	✓
UC-6	✓	✓	✓	✓

Source. Own elaboration, 2025.

4.4. Discussion

Across the six use cases and all configurations, the Entities Small Language Model (SLM) consistently surfaced discriminative terms that resolved to the correct tables in DataHub. google/gemma-2-2b-it performed well enough to be indistinguishable from both its Expanded counterpart, google/gemma-3-4b-it and Gemini 2.5 PRO, therefore, it seems that this small task is perfectly handled with a Compact SLM despite the mix of Spanish and English across columns and descriptions.

For SQL query generation, differences start to occur. The SQL role succeeded on 4/6 tasks with the Compact-SLM Stack (UC-1/2/3/4) and on 5/6 with the Expanded-SLM Stack (UC-1/2/3/4/5). The Gemini 2.5 PRO baseline completed 6/6.

When comparing the one-table queries, we find that the Compact-SLM SQL LLM powered by Qwen/Qwen2.5-Coder-3B-Instruct handles the load well enough with optimal queries with minor

issues such as the lack of an ORDER BY clause in UC-2 and UC-4. Its bigger sibling, Qwen/Qwen2.5-Coder-7B-Instruct, is also outputting correct queries, but they tend to feature heavier idioms (e.g., unnecessary aggregation), yielding a ✓*, instead as well as a lack of ordering. Meanwhile, Gemini outputs optimal queries with additional, not-asked for orders that do improve usability.

For multiple tables, Qwen/Qwen2.5-Coder-3B-Instruct, cannot generate accurate queries as it fails UC-5 and UC-6. For UC-5, it does output a compilable query that returns the correct cruiseship and "so2 media horaria", however, it takes the hour from the arrival table instead of the actual measurement time from the air-quality table. Here, the Expanded-SLM Stack SQL LLM does work this out and outputs the correct hours, although it uses a non-sargable day filter where a half-open range would be cheaper, which Gemini uses to further optimize the query. At UC-6, however, even Qwen/Qwen2.5-Coder-7B-Instruct cannot output the correct query as its prediction features a GROUP BY alias misuse causing a compile error as it also does for the Compact configuration. Gemini 2.5 PRO correctly uses the islands-and-gaps pattern (ROW_NUMBER() offsets + COUNT(*)≥5) to find truly consecutive high-SO_2 runs and then performs a proper overlap join to docked cruises and computes berth-period averages while also yielding a computationally optimal plan under the rubric.

A key aspect of our architecture here is the lack of fixing a not-compilable query with EXPLAIN and a subsequent prompt. Neither configuration actually was able to fix the prompt to compile the failed UC-6 SQL query, which may indicate that its usage on SLMs is still not as useful as in LLMs, although, it does no harm to retry once we know the query does not compile.

The Domain-Expert role was the main source of invalid system-wide outputs for SLMs: Compact yielded grounded explanations in 2/6 tasks (✓* in UC-1/3), Expanded in 4/6 (✓* in UC-1/3/4/5), while Gemini produced **6/6** correct, contextual explanations. For UC-2, both SLM configurations misunderstood the SQL output and mixed daily means with hourly values due to column naming and the lack of a specific column with the daily mean in the output. This could be addressed in the future with more rules to always display discriminatory values such as means. For UC-4, the Compact Domain Expert misread the arrival time column and incorrectly stated that all ships arrived the same day, although two arrived the 22nd and one arrived the 21st. Though a minor error, it does confuse the end user. Expanded did not make that mistake and for UC-1/3/4/5, it correctly summarized and re-stated valuable information in a natural language manner. However, neither configuration was capable of giving insights on their own using the context provided (general role information and columns descriptions), unlike Gemini, which tried to establish correlations and contextualize the results for non-experts end-user.

In the end, the Compact-SLM configuration proved valid for data discovery and simple queries but lacked more complex explanations. The Expanded-SLM Stack could offer more robust queries and explanations but still cannot process the top more complex queries at the level of frontier models. As a discovery assistant, both configurations and especially the Expanded one works well today: it can find relevant tables, compose and validate SQL, and summarize findings quickly. However, as is the case with LLM queries pushing the limits, results cannot be blindly trusted. Nonetheless, it may be much faster to use the assistant to obtain the desired information and then verify it than having to look for it out-of-the-box. For that, it can be an asset for non-technical users that want an easier way to interact with the database, and it can be quite useful for technical personnel to use as a baseline.

4.4.1. Practical Implications, Study Limitations, and Implementation Risks

Our SLM-powered assistant offers significant potential out-of-the-box for public administrations, as seen with the SENSE Citiverse for the city of Cartagena. It directly addresses the challenge of making vast datasets not only accessible but also inspectable to non-technical staff through natural language. In turn, workloads could be reduced across all the employees, allowing personnel to focus on more complex, high-value tasks, which should be the goal of Gen AI systems across the board, but specially for public service applications.

However, both the current state of the solution and the study itself have limitations. The study is based on domain-specific use cases and data, which cannot completely cover all the

spontaneous user queries scenarios. In order to generalize the usability of the solution, a large-scale user study should be performed to retrieve human-feedback on crucial metrics like trust, ease of use, and overall correctness and user satisfaction as well as introducing new domains in which to deploy the assistant.

Finally, deploying such a system comes with several implementation risks, with data privacy and security at the top of the priority list. Although a self-hosted model allows for clear data sovereignty and increased security control on access, the system still requires clear protocols and user-access policies, particularly if data-ingestion is included as a functionality down the road. Therefore, proactive security testing, or red-teaming, is essential to identify these and other vulnerabilities. There is also the risk of algorithmic bias and fairness, as AI models can amplify biases present in their training data, potentially leading to inequitable service outcomes (Cortés-Cediel et al., 2023). Furthermore, a successful implementation would require some degree of organizational readiness. To mitigate the previous risks and leverage the system at its fullest, staff should be trained on both how to use our front-end application and prompting techniques to ensure optimal utilization. In addition, technical staff should be availed to maintain the system and integrate it with existing systems.

5. Conclusions

European data spaces aim to widen access to public and industrial data, and they now reach municipal operations at scale. Yet, users still have to navigate these resources with precision. Cartagena's SENSE Citiverse exemplifies this gap: technicians must assess air-quality telemetry and even correlate it with operational data from the port and the city, but writing particular SQL queries to assess heterogeneous data sources can be very time-consuming, specially for non-technical users. Natural Language (NL) interfaces can reduce this barrier if they generate faithful queries and return grounded explanations.

This work addresses that need with a compact, on-premise system that converts NL to SQL and answers in NL while preserving data sovereignty and cost control. With three cooperating Small Language Models (SLMs) filling in Large Language Models (LLMs) roles: 1) Entities LLM; 2) SQL LLM and 3) Domain Expert LLM, this approach advances human-centered access while respecting sovereignty, cost, and sustainability constraints that often preclude dependence on remote LLMs.

The architecture paired SLMs with strong open source tooling: DataHub for data and metadata discovery, Trino for SQL execution, and KServe for reliable serving and autoscaling. This separation of concerns reduced prompt length, limited token budgets, and made failures observable and recoverable. The design also aligned with recent evidence that multi-agent pipelines improve text-to-sql robustness through decomposition and database feedback.

The pipeline is simple: First, the Entities LLM extracts structured terms from the user's NL prompt. Second, the orchestrator queries DataHub to ground those terms to concrete tables and columns. Third, the SQL LLM composes a query which Trino runs; on error, the orchestrator retries with the compiler feedback and then executes the repaired query. And, finally, the Domain Expert LLM reads the question, the validated SQL, the result set and its SQL context, and produces a concise, domain-aware explanation. KServe hosts each role as an autoscaled endpoint with revision control and metrics

For the evaluation, two SLM stacks, Compact (≤ 3 Billion parameters) and Expanded (≤ 8 Billion parameters) were deployed, revealing clear strengths and limits. Regardless of stack, the Entities LLM consistently emitted discriminative terms that resolved to the correct tables, despite these tables and metadata mixing Spanish and English. The Compact Stack SQL generation capabilities excelled in one table queries, with the Expanded stack adding support for cross table queries; but both fell short on the more complex temporal-join case where the baseline Gemini 2.5 Pro succeeded. The Domain Expert SLM produced faithful but sparse interpretations in simpler cases and lost grounding on complex outputs, in particular for the Compact Stack, while the Expanded could effectively explain the SQL output even in more complex scenarios.

In sum, a compact, sovereign stack can deliver fast, useful text-to-sql for municipal telemetry and operations. The system lowers the barrier for non-experts, scales horizontally, and preserves privacy by design. Its modularity invites incremental upgrades while containing operational risk.

5.1. Future work

This paper validates the proposed architecture using two specific configurations of models on on-cloud hardware. However, future work will evaluate performance across multiple model and hardware configurations to establish clear benchmarks for real-world adoption, particularly on consumer-grade and enterprise-level hardware. The objective of this future research will be to quantify the hardware-related trade-offs, therefore, three distinct tiers of model configurations are proposed: Compact (≤ 4 GB VRAM), Balanced (8–16 GB), and Expanded (≥ 24 GB).

For each configuration, performance will be measured against critical metrics, including the inference time (latency) of each individual SLM, the total end-to-end latency from prompt submission to final response, and the token generation rate (tokens/second). The overarching goal is to identify configurations that can reliably deliver a response in under 10 seconds, ensuring a fluid and interactive user experience.

Experiments will compare prompt-only vs. lightweight fine-tuning for the SQL and Domain Expert roles. Extended use cases will also be considered to further validate the proposed architecture across different scenarios, with specific testing per SLM as well as end-to-end.

A new ingestion-with-LLM pipeline will enrich metadata for new tables. The Domain Expert SLM will generate ~200-token, domain-specific descriptions per column from sample rows and naming conventions. Then, the Orchestrator will perform the data ingestion through Trino and upload field descriptions via DataHub. Also, as introduced, a Spanish translator actor will enable Spanish–English prompts to better exploit the SLMs when the prompt is in Spanish, which is key for Cartagena adoption. With the extended experiment results as well as new functionalities, a foundation can be set for further user studies to retrieve critical human-feedback.

The current roadmap aims to both evaluate the current architecture, expand its functionality and specialize its knowledge to Cartagena's SENSE Citiverse specific roles while keeping the framework modular and open for future projects. If successful, this assistant can serve as an EU-compliant template for NL interfaces across dataspaces which satisfies AI Act and emerging AI-related regulations in Europe to make this type of models suitable for the European Digital Economy, with a special focus on Digital Transformation for Smart Cities; this model would be available into the EDIC Marketplace to make it available as Open Source to be used together with European Local Digital Twins, compatible with the EU LDT Toolbox, and into sandboxes as Citcom.AI TEF Sandbox.

References

Basant, A., Khairnar, A., Paithankar, A., Khattar, A., Renduchintala, A., Malte, A., Bercovich, A., Basant, A., Khairnar, A., Paithankar, A., Khattar, A., Renduchintala, A., Malte, A., Bercovich, A., Hazare, A., Rico, A., Ficek, A., Kondratenko, A., Shaposhnikov, A., Bukharin, A., Taghibakhshi, A., Barton, A., Mahabaleshwarkar, A. S., Shen, A., Tao, A., Guan, A., Shors, A. (…) Chen, Z. (2025). NVIDIA Nemotron Nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *ArXiv*, *V4*. https://arxiv.org/abs/2508.14444

Belcak, P., Heinrich, G., Diao, S., Fu, Y., Dong, X., Muralidharan, S., Lin, Y.-C., & Molchanov, P. (2025). Small language models are the future of agentic AI. *ArXiv*, *v2*. https://arxiv.org/abs/2506.02153

Chen, A., Bundele, M., Ahlawat, G., Stetz, P., Wang, Z., Fei, Q., Jung, D., Chu, A., Jayaraman, B., Panth, A., Arora, Y., Jain, S., Varma, R., Ilin, A., Melnychuk, I., Chueh, C., Sil, J., & Wang, X. (2025). Text-to-SQL for enterprise data analytics. *Workshop on Agentic AI for Enterprise at KDD '25*, Toronto, ON, Canada. https://arxiv.org/abs/2507.14372

Cortés-Cediel, M. E., Segura-Tinoco, A., Cantador, I., & Rodríguez Bolívar, M. P. (2023). Trends and challenges of e-government chatbots: Advances in exploring open government data and citizen participation content. *Government Information Quarterly*, *40*(4), 101877. https://doi.org/10.1016/j.giq.2023.101877

DataHub Project. (2015a). *DataHub GraphQL API*. DataHub. https://docs.datahub.com/docs/api/graphql/overview

DataHub Project. (2015b). *DataHub quickstart guide*. DataHub. https://docs.datahub.com/docs/quickstart/

DataHub Project. (2024). *DataHub APIs and SDKs overview*. DataHub. https://docs.datahub.com/docs/api/datahub-apis

DataHub Project. (2025). *DataHub | Modern data catalog & metadata platform*. DataHub. https://datahub.com/

Google. (2025a). *Gemma-2-2b-it* [Large language model]. Hugging Face. https://huggingface.co/google/gemma-2-2b-it

Google. (2025b). *Gemma-3-4b-it* [Large language model]. Hugging Face. https://huggingface.co/google/gemma-3-4b-it

Hong, Z., Yuan, Z., Zhang, Q., Chen, H., Dong, J., Huang, F., & Huang, X. (2024). A next-generation survey of LLM-based text-to-SQL database interfaces [Preprint]. *arXiv*. V8. https://arxiv.org/abs/2406.08426

Inglés-Romero, J. F., Ferri, M., & Jara, A. J. (2025). Exploring human usability challenges in dataspaces. En J. Theissen-Lipp, P. Colpaert, A. Pomp, E. Curry & S. Decke (Eds.) *The Third International Workshop on Semantics in Dataspaces, ESWC 2025*, Portorož, Slovenia.

Jiang, Y., Pang, P. C.-I., Wong, D., & Kan, H. Y. (2023). Natural language processing adoption in governments and future research directions: A systematic review. *Applied Sciences*, *13*(22), 12346. https://doi.org/10.3390/app132212346

KServe Project. (2024). *modelmesh: Distributed model serving framework* (Version 0.12.0) [Computer software]. GitHub. https://github.com/kserve/modelmesh

KServe Project. (2025a). *Control plane*. KServe Documentation. https://kserve.github.io/website/docs/concepts/architecture/control-plane

KServe Project. (2025b). *Deploy your first GenAI service*. KServe Documentation. https://kserve.github.io/website/docs/getting-started/genai-first-isvc

KServe Project. (2025c). *Deploy your first predictive AI service*. KServe Documentation. https://kserve.github.io/website/docs/getting-started/predictive-first-isvc

KServe Project. (2025d). *KServe*. Retrieved October 3, 2025, from https://kserve.github.io/website/

KServe Project. (2025e). *kserve: Standardized distributed generative and predictive AI inference platform for scalable, multi-framework deployment on Kubernetes* (Version 0.15.2) [Computer software]. GitHub. https://github.com/kserve/kserve

Kubeflow. (2025, July 31st). *Introduction: A brief introduction to KServe*. Kubeflow Documentation. https://www.kubeflow.org/docs/components/kserve/introduction/

Kubernetes. (2024, April 20th). *Kubernetes documentation*. https://kubernetes.io/docs/home/

Meta AI. (2024a). *Llama-3.1-8B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/meta-llama/Llama-3.1-8B-Instruct

Meta AI. (2024b). *Llama-3.2-3B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/meta-llama/Llama-3.2-3B-Instruct

Qwen Team. (2025a). *Qwen2.5-Coder-3B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/Qwen/Qwen2.5-Coder-3B-Instruct

Qwen Team. (2025b). *Qwen2.5-Coder-7B-Instruct* [Large language model]. Hugging Face. https://huggingface.co/Qwen/Qwen2.5-Coder-7B-Instruct

SENSE Project. (2025a, June 19th). *CitiVerse*. Senseverse. https://senseverse.eu/citiverse/

SENSE Project. (2025b, June 13rd). *Pilots*. Senseverse. https://senseverse.eu/pilots/

SENSE Project. (2025c, June 5th). *Senseverse | Building smart, connected citizens with SENSE*. Senseverse. https://senseverse.eu/

Shi, L., Tang, Z., Zhang, N., Zhang, X., & Yang, Z. (2025). A survey on employing large language models for text-to-SQL tasks. *ACM Computing Surveys, 58*(2), Article 54. https://doi.org/10.1145/3737873

Trino. (2020). *EXPLAIN*. Trino Documentation. https://trino.io/docs/current/sql/explain.html

Trino. (2025a). *Concepts*. Trino Documentation. https://trino.io/docs/current/overview/concepts.html

Trino. (2025b). *Connectors*. Trino Documentation. https://trino.io/docs/current/connector.html

Trino. (2025c). *EXPLAIN ANALYZE*. Trino Documentation. https://trino.io/docs/current/sql/explain-analyze.html

Ukil, A., Jara, A., Gama, J., & Bellatreche, L. (2025). Buck the trend: Make LLMs specific and reduce the cost of intelligence. *28th European Conference on Artificial Intelligence*, Bologna, Italy.

Yigitcanlar, T., David, A., Li, W., Fookes, C., Bibri, S. E., & Ye, X. (2024). Unlocking artificial intelligence adoption in local governments: Best practice lessons from real-world implementations. *Smart Cities, 7*(4), 1576–1625. https://doi.org/10.3390/smartcities7040064