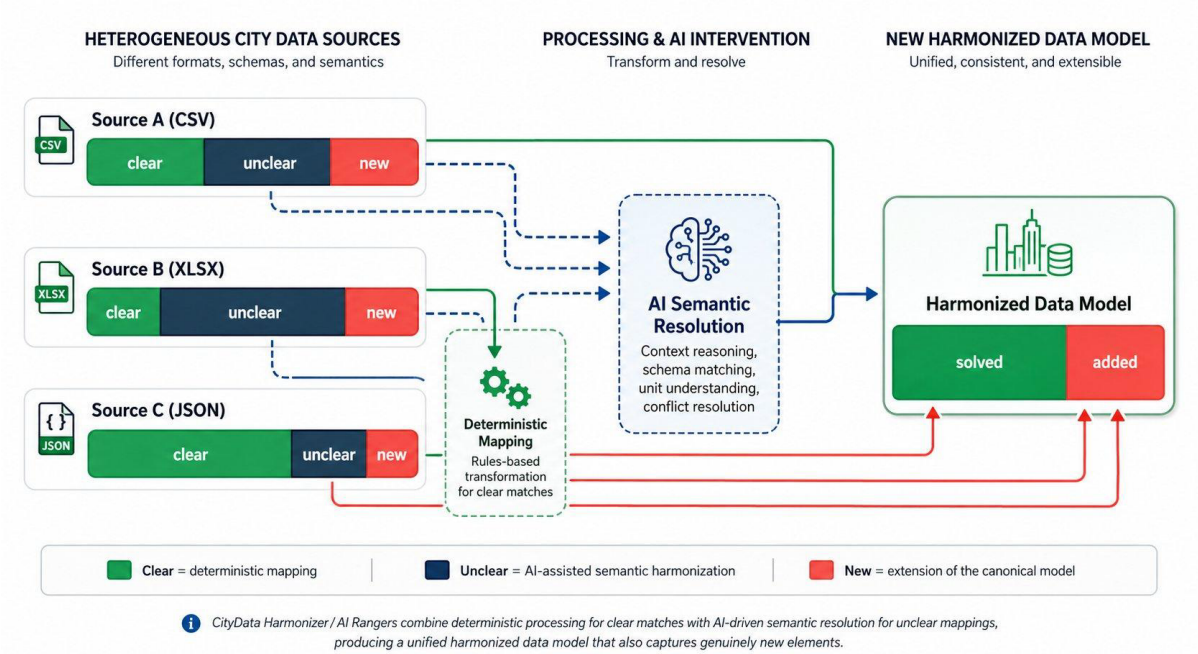
CityData Harmonizer: AI Rangers for City Data Harmonization

Findings from the MIMathon 2026 | Porto Use Case 3 (Energy Data Harmonization)
Team Members: Marwen Chaabouni, Mohamed Benkedim, Olaf Gerd Gemein
Date: 1.6.2026

The Challenge

Cities are drowning in data, but the data does not fail because it is missing. It fails because meaning and format fragments across systems. In the Porto energy use case, the same quantity appears as energy_kWh, consumption_kWh, and energyConsumed across providers: one concept, a dozen names, zero compatibility. Files arrive as CSV, XLSX, and JSON, with different delimiters, encodings, and shapes. Without a shared canonical target, comparing or aggregating across providers is impossible at scale.

The task is therefore not a simple 1:1 field match. It is, first, a semantic problem: take several non-interoperable datasets within one domain, identify the right canonical target, fill the gaps, and produce one combined output that carries all content from every source without losing provenance.



Clear matches are handled by deterministic mapping; unclear ones by AI semantic resolution; genuinely new elements extend the canonical model. The result is one unified, consistent, extensible data model.

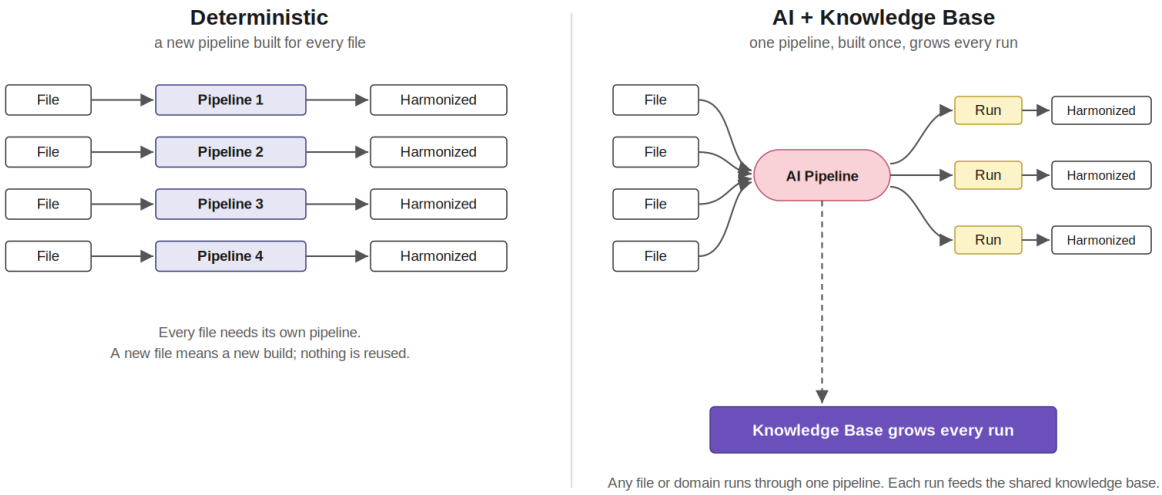
We did not begin by asking whether to use AI. In an era where AI is reshaping how every kind of work gets done, our first question for the Porto challenge was how to solve it with AI, not whether to. It was only when we presented our solution for the first time that we met the question we had skipped over: "*Why use AI at all, when a deterministic system already solves this exact problem?"* It is a fair question. The honest answer is in what the system does in practice.

Picture the work as it is done today. A city receives a new energy file, and an engineer writes a dedicated script to clean and map it. The next file, in a different shape, needs another script; a file from another domain needs a different pipeline entirely. Every new source is a new build, and the effort never compounds.

Our system inverts this. The pipeline is built once and run many times, so a new file is a new run, not a new pipeline. Every run also teaches the system, because each confirmed mapping is written to a knowledge base that the next run reuses, so the work compounds instead of resetting. And because the logic reasons about meaning rather than a fixed schema, the same pipeline accepts files from a domain it has never seen, which is why our energy pipeline harmonized water-quality data without a single code change.

The payoff is less manual work, decisions made across far more data, and knowledge that carries from one city and one domain to the next. The same reasoning extends naturally further still: where a source writes "kilowatt"

and the canonical model stores “kW”, an AI step can notice the mismatch and reconcile it, whereas a deterministic system would simply error out until a human encodes the rule by hand.



Deterministic harmonization builds one pipeline per file. The AI approach builds one pipeline, runs it on any file or domain, and grows a shared knowledge base with every run.

This is the difference that the rest of this document builds on. A deterministic ETL pipeline can execute known mappings, but it cannot explore an unknown target schema, infer plausible mappings from context, or decide which source is authoritative when sources conflict. So the deeper question we set out to answer was never why use AI, but how: *how to make the harmonization process **replicable, reusable, and domain-agnostic**.*

The Rangers Solution

The CityData Harmonizer is a nine-act pipeline of agents we call the Rangers, trained in the spirit of the OASC Academy as MIM champions. Its founding principle is a strict separation between probabilistic proposal and deterministic execution: the AI rangers propose, the deterministic rangers verify and execute, and nothing reaches production data without passing a rule-based quality gate.

Four AI-powered rangers handle investigation and judgment. Five deterministic rangers, which use no LLM, handle parsing, verification, execution, validation, and lifecycle. The roster:

Ranger	Type	Role in the pipeline
SchemaSelector	AI	Runs pre-pipeline. Discovers the best-fitting OASC Smart Data Model for the source via the Smart Data Models MCP server.
Adapter	Deterministic	Reads any input format (CSV, XLSX, JSON) and turns it into a clean list of records. Pluggable: new formats need no pipeline changes.
Profiler	AI	Reads the source plus its manifest metadata and produces a Source Intelligence File: field names, types, distributions, quality hints.
Planner	AI	Maps source fields to canonical targets and produces a Harmonization Plan, routing each field by confidence score.
Resolver	AI	Takes the uncertain mappings the Planner deferred, runs deep analysis, and returns a decision per field: INFERRED, UNMAPPED, or REVIEW.
Knowledge Engine	Deterministic	The quality gate. Runs four rule-based checks and returns APPROVE, REJECT, or ESCALATE. No LLM involved.
Executor	Deterministic	Applies the approved plan row by row using nine pure transform functions. Writes provenance for every action.
Validator	Deterministic	Checks every output record against the canonical JSON Schema. Returns pass or fail with error messages. Failures block publication.
PromotionManager	Deterministic	Manages the Knowledge Base lifecycle: candidate, reviewed, approved, promoted.

Confidence routing is explicit and deterministic. Above 0.85, a mapping is auto-approved as INFERRED; between 0.60 and 0.84, it is sent to the Resolver as UNCERTAIN; below 0.60, it is excluded as UNMAPPED and surfaced for human review. The canonical target model is selected from or extended within the OASC Smart Data Models repository[1], and the SchemaSelector discovers it automatically through the Smart Data Models MCP server[2], which gives the agent the exact JSON-LD and NGSI-LD structures it needs and so reduces hallucination.

This work also surfaced a real gap in the catalogue. Today, no Smart Data Model captures observed energy consumption across providers and fuel types. ACMeasurement is electricity-only and built for phase-level electrical engineering, not city-wide reporting. ConsumptionCost supports only monthly granularity, uses free-text energy types with no controlled vocabulary, and carries a currency enumeration typo that has never been fixed. EnergyConsumer is a grid-topology object for power-flow simulation, not a metered observation. None of them can hold a quarterly gas bill, an hourly district-heating reading, and an annual electricity benchmark in the same schema, and none support prosumer flows, carbon reporting, weather normalization, or data-quality flags.

We reviewed seven city open-data portals, five international standards (ESPI / Green Button, IEC 61968-9, CityGML Energy ADE, ISO 52000, EPBD), and every existing SDM energy entity. EnergyConsumptionObserved fills it: one entity for any fuel at any granularity, with eight mandatory fields and 41 in all, fully aligned to existing standards and designed from day one for AI-driven harmonization across heterogeneous city data sources. We propose it back to the Smart Data Models community as the canonical target for city energy reporting[3].

Two properties follow from this design. First, deterministic reproducibility: the same Harmonization Plan applied to the same data yields identical output every time, because the AI decisions are isolated from execution. Second, self-learning convergence: every confirmed mapping is stored in the Knowledge Base, so when the Planner later finds a promoted entry for a field, it skips the LLM call entirely. For stable, recurring sources, the LLM cost converges to zero, and every run is faster than the last.

The Datasets We Tested

To test the pipeline, we deliberately chose datasets that did not agree with each other. Within the energy domain, we took five independent sources with different owners, file formats, field names, and levels of granularity, ranging from a single household meter to country-level aggregates, and harmonized them all to one canonical model, EnergyConsumptionObserved. We then repeated the exercise in a second domain, water quality, with no changes to the pipeline, to test the claim that the approach is domain-agnostic.

Energy domain: five heterogeneous sources, one canonical target.

Dataset	Format	Source	What it tested
Fronius Energy Consumption, City of Maia	XLSX (private)	Pedro C. C. Pimenta, GitHub	A real proprietary monitoring export, and the only XLSX and only private source. Closest to a live city feed. Carries directly-consumed, total, and grid-drawn energy.
Tetouan Smart Grid, 10-minute Power Consumption	CSV	Tetouan Smart Grid Research	Power reported across three separate zones with 10-minute timestamps. Tested temporal harmonization and multi-zone structure mixed with weather fields.
Electricity Consumption with Environment Variables	CSV	Aamir Ansari, Kaggle	Household-level electrical detail (active and reactive power, voltage, intensity, sub-metering) plus environmental variables. A fine-grained meter schema.

[1] Smart Data Models initiative, co-governed by OASC together with FIWARE Foundation, TM Forum, and IUDX. *https://smartdatamodels.org/*

[2] Smart Data Models MCP Server, a reference implementation connecting LLM agents to the Smart Data Models ecosystem via the Model Context Protocol (A. Galdemas, agaldemas/smartdatamodels-mcp). *https://smartdatamodels.org/index.php/new-mcp-service-to-use-smart-data-models-in-your-local-ai-agent-autonomous-light-management-4-smart-data-models/*

[3] "EnergyConsumptionObserved" Data Model (own work) https://drive.google.com/file/d/18sdZUOS8tDuzD1_Hos-cCP9tCfsAhWSo/view?usp=sharing

Dataset	Format	Source	What it tested
Climate and Energy Consumption, 2020 to 2024	CSV	Emirhan Akku, Kaggle	Country-level aggregation with emissions, renewable share, pricing, and demographics. A very different granularity and breadth from meter data.
Smart Energy Consumption and Peak Load	CSV	Jay Joshi, Kaggle	Building characteristics, occupancy, and peak-load indicators. Tested building and occupancy context alongside raw consumption.

Water-quality domain: two sources, harmonized with no pipeline changes.

Dataset	Format	Source	What it tested
Water Quality Monitoring (Source A)	CSV	Pratiksha827, GitHub (open)	First of two water sources. Tested mapping to the WaterQualityObserved Smart Data Model with no change to the pipeline.
Water Potability (Source B)	CSV	A. Kadiwal, Kaggle, mirrored by Sarthak-1408 (CC0)	A second water source with different fields. Tested merging two non-interoperable water datasets into one canonical model.

Across both domains, the result was the same: heterogeneous, non-interoperable inputs converged to a single Smart Data Model output. The energy datasets prove multi-source convergence within one domain. The water datasets prove the same pipeline carries to a new domain and a new canonical model without code changes.

How the Rangers Satisfy the MIMs

MIM compliance is not a checklist applied after the fact. Each mechanism is satisfied by a specific ranger at a specific stage, which is why the table below names the responsible ranger for every requirement.

MIM	Requirement	How it is satisfied	Responsible Ranger
MIM1	Unique, persistent identifier per entity	Auto-generates urn:ngsi-ld:EnergyConsumptionObserved:<uuid> for every harmonized record.	Executor
MIM1	Cross-system entity linking	refMeter and refBuilding fields carry NGSI-LD URNs linking each observation to its physical meter and building.	Executor
MIM1	Semantic typing of entities	Every record carries an explicit type aligned to NGSI-LD.	Executor
MIM1	Ontology-level mapping across sources	Heterogeneous field names are mapped to canonical equivalents with confidence scores and persisted evidence.	Profiler, Planner, Resolver
MIM2	Models explicit, documented, unambiguous	The schema defines all attributes with types, units, enumerations, and references to schema.org and OASC SDMs.	SchemaSelector, Planner
MIM2	Build on standardized community models	The canonical schema composes OASC Smart Data Models via allOf/$ref (GSMA-Commons, Location-Commons, EnergyConsumption).	SchemaSelector
MIM2	Cross-model transformation to common model	The full nine-act pipeline transforms any provider schema into one canonical model automatically.	Planner, Executor
MIM3	Adequate machine-readable metadata	Each source manifest is a structured descriptor: owner, domain, format, columns, units, and quality notes.	Profiler
MIM3	Traceable provenance and trust lineage	A provenance record captures full lineage per record: source row, mapping IDs, every transform, KE verdict, and validation result.	Executor, Validator
MIM7	Geospatial data in open standards	combine_lat_lon and parse_geojson transforms produce OGC-compliant GeoJSON Point geometry.	Executor
MIM7	Contextual datasets comply with MIM1 and MIM2	Location is embedded in the same NGSI-LD record carrying the MIM1 URN and MIM2 canonical schema.	Executor

A Method for Embedding MIMs into an Agentic Pipeline

The table above shows that the Rangers satisfy the MIMs. The more useful finding is the method behind it, because it is the part any team can reuse. Embedding a MIM into an agent is not a matter of writing the rule into a prompt and trusting the model to obey it. A prompt is not a guarantee. What we found is that a MIM can be split into parts that a system can enforce, and that the split is already handed to us by the way MIMs are written.

Every MIM is documented against the ITU Y.4505 standard[4], so every MIM carries the same three working parts: a requirement (what must be true), a mechanism (how it is achieved), and a conformance test (how you check it was achieved). Whereas a requirement is a judgment about meaning, a mechanism is a fixed procedure, and a test is a pass-or-fail check. Our pipeline already has one home for each kind of work, so embedding a MIM means sending each part to its designated home.

Part of the MIM	What it is	Where it goes	Why it goes there
Requirement	A judgment about meaning that changes from one source to the next	An AI ranger, as a proposal, with a confidence score	Meaning cannot be fixed in advance; it has to be inferred for each source
Mechanism	A fixed, repeatable procedure	A deterministic ranger, as a pure transform that is logged	It must run identically every time and stay auditable
Conformance test	A pass-or-fail check	The Knowledge Engine, as a gate rule	Compliance must be proven before a record is published, not assumed

MIM1 shows all three at work. Its requirement, mapping a source field to the right canonical field, is a judgment, so it goes to the Profiler, Planner, and Resolver, which propose a mapping with a confidence score. Its mechanism, which gives every record a unique and persistent identifier, is a fixed procedure, so the Executor mints the URN the same way every time. Its conformance test, that every record must carry a valid identifier and type, becomes a rule in the Knowledge Engine that rejects any record without one. One MIM, split three ways, each part handled by the layer best suited for it.

Read as a pattern, the rule is simple: the AI layer answers the MIM's open questions, the deterministic layer runs its fixed mechanisms, the Knowledge Engine checks its conformance tests, and the Knowledge Base remembers the result. A new MIM is added by reading its three Y.4505 parts and sending each to its home, never by rewriting the pipeline. That is what makes the approach replicable across MIMs and domain-agnostic across use cases.

The same four steps that satisfied MIM1, MIM2, MIM3, and MIM7 today extend cleanly to the MIMs we have not yet implemented. Each one is simply a new requirement to classify, decompose, assign, and promote, and for several of them, the integration point is already clear. This is the roadmap the method produces:

MIM	What it requires	How does the same method extend the system
MIM0	Accessible, standardized, interoperable APIs; no data silos	Push harmonized NGSI-LD records to an Orion-LD or Scorpio context broker after execution. The harmonizer becomes an ETL feeder and downstream systems query one endpoint regardless of original provider; MIM0 compliance is inherited from the broker.
MIM2	Transformation rules must be reusable and self-learning	Promote the Knowledge Base from a per-deployment store into a shared community registry, so a mapping confirmed in one city is reusable by another. This is the self-expanding behaviour already in the pipeline, scaled across municipalities.
MIM3	Governance defined and comprehensible; assets discoverable; agreements enforceable	Add a governance block to the manifest (licence, usage terms, category), expose manifest metadata as a DCAT-AP catalogue endpoint, and integrate a machine-readable data-sharing policy (IDS or GAIA-X) enforced at the API gateway.

[4] ITU-T Y.4505, "Minimal interoperability mechanisms (MIMs) for smart cities and communities," the international standard that documents each MIM against a common skeleton of objective, capabilities, requirements, mechanisms, and conformance tests, aligned with the OASC MIMs framework. https://oascities.org/minimal-interoperability-mechanisms/

MIM	What it requires	How does the same method extend the system
MIM4	Individuals can control their personal data	When the system scales down to individual household meters, add a consent-management step (MyData or Solid pattern) before any personal consumption record is harmonized.
MIM6	Identity lifecycle and cryptographic data integrity	Add OAuth2 / JWT authentication and TLS to the API layer with role-based access per provider and consumer, and sign harmonized output and provenance files, aligned to ISO 27001 and NIS2.
MIM7	CRS compliance and INSPIRE metadata for the EU context	Declare crs EPSG:4326 on all GeoJSON output and add INSPIRE-compliant metadata fields to the manifest schema, extending the geospatial transforms already in place.
MIM8	Local Digital Twin layers cover pre-processing, analysis, and cross-sector collaboration	Formally declare the harmonizer as the LDT layer-3 pre-processing component, feed its UTC-normalized time series into demand-forecasting and Climate Contract tracking, and extend ingestion to mobility, water, and waste so one pipeline serves a multi-sector city twin.

None of these requires a new pipeline. Each is the four-step method applied once more: classify the MIM, decompose it into proposal, skill, and test, assign each part to the right layer, and promote the result so the next city inherits it.

What We Recommend Doing with the MIMs Next

The question put to us at the MIMathon was how to make the MIMs machine-readable, so that an AI system can use them to reason about a real problem. We did not stop at an answer on paper. We built a working pipeline that reads the MIMs' structure and enforces it, tested it across two domains, and proposed a new data model back to the community. What follows is therefore not a theory we are asking others to explore. It is a result we have already demonstrated and are ready to take further.

Our recommendation is to give every MIM a machine-readable companion. The MIM text stays as it is, written for people; alongside it sits a structured version, derived from the Y.4505 parts each MIM already has, that an agent can consume directly. Our pipeline already does this in practice for MIM1, MIM2, MIM3, and MIM7, so we can produce the first reference example now and let the others follow the same pattern.

What we are handing over today is the groundwork: the open-source CityData Harmonizer, the method behind it, the glossary, the presentation, the proposed EnergyConsumptionObserved model, and this document. The code runs end-to-end on the Porto energy use case. It is open for anyone to inspect, and it is the proof that the approach works rather than a promise that it might.

Turning that groundwork into a life service cities can rely on is real daily work , actually at least three pieces of work stand between the prototype and production:

1. **Harden the pipeline for production.** Standards-compliant NGSI-LD output, live model selection through the Smart Data Models MCP server, and an expert-in-the-loop interface for the stewards who confirm mappings.
2. **Build the shared skill and transform the library.** A versioned registry where a mapping confirmed in one city becomes reusable by every other city. This is what makes MIM compliance compound across the network instead of restarting in each city. The shared skill (-set) we envision is the new home for the MIMs: “MIM-AI”.
3. **Run a pilot with founding cities.** A small group of member cities runs the harmonizer on real data, while the OASC Academy trains the first stewards, the AI Rangers, who operate the system, thus feed and grow the shared knowledge library.

We have built the foundation, shown it works, and we are ready to contribute or kind of coordinate this work within OASC. The remaining part is to turn the proven prototype into a production service for all member cities and implement all the necessary improvements, which is a task our team can lead with some pilot founding cities: probably the MIM Champions to start with.

Make the MIMs machine-readable, prove it on an open pipeline that already exists, and let a shared library carry the knowledge from one city to the next, and OASC positions itself at the forefront of data interoperability worldwide.