National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science and Technology

Architectural Frameworks for extracting semantic and spatial references from unstructured text using AI: The case of Energy markets

DIPLOMA THESIS

by

Antonis Agoris

Supervisor: Vassilios Vescoukis,
Professor, NTUA

Athens, September 2025

National Technical University of Athens
School of Electrical and Computer Engineering
Division of Computer Science and Technology

Architectural Frameworks for extracting semantic and spatial references from unstructured text using AI: The case of Energy markets

DIPLOMA THESIS

by

Antonis Agoris

Supervisor: Vassilios Vescoukis,
Professor, NTUA

Approved by the three-member scientific committee on 7th November 2025.

Vassilios Vescoukis,	Nikolaos Papaspyrou,	Aris Dimeas,
Professor, NTUA	Professor, NTUA	Asst. Professor, NTUA

Athens, September 2025

Antonios Agoris
Graduate of School of Electrical and Computer Engineering, National Technical University of Athens

Dedication

To my family,

who, with patience, sacrifice, and quiet love,
offered me a place to stand so that I could learn, grow,
and take my first true steps into this work.

Archimedes once said:
«Δός μοι πᾶ στῶ καὶ τὰν γᾶν κινάσω» —
“Give me a place to stand, and I will move the earth.”

In my life, that place has never been a point in space,
nor a lever of iron,
but the living support you have given me:
your trust when I hesitated,
your encouragement when I doubted,
and your willingness to make room for me
even when it came at a cost.

This work is offered as a small fruit
of that shared love, strength, and faith.

Περίληψη

Η παρούσα διπλωματική εργασία προτείνει ένα αρχιτεκτονικό πλαίσιο για την εξαγωγή σημασιολογικών και χωρικών αναφορών από αδόμητο κείμενο με χρήση Τεχνητής Νοημοσύνης, με εφαρμογή στον τομέα των ενεργειακών αγορών. Ο ευρωπαϊκός ενεργειακός τομέας ψηφιοποιείται ραγδαία, παράγοντας μεγάλα σύνολα ανοικτών και ημι-ανοικτών δεδομένων, όπως δείκτες αγοράς (ENTSO-E), μετρήσεις συστημάτων μεταφοράς (ΑΔΜΗΕ), επίπεδα (layers) γεωχωρικών δεδομένων για υποδομές και μετεωρολογικά πεδία. Ωστόσο, τα δομημένα μετρητικά δεδομένα από μόνα τους σπάνια επαρκούν για να εξηγήσουν ανωμαλίες—όπως αιφνίδιες αυξήσεις τιμών ή μεταβολές στο μίγμα παραγωγής—των οποίων τα αίτια αποτυπώνονται κυρίως σε αδόμητες πηγές κειμένου, όπως ειδησεογραφία και ανακοινώσεις πολιτικής.

Το προτεινόμενο σύστημα ενσωματώνει δομημένες χρονοσειρές με γνώση που εξάγεται μέσω Τεχνητής Νοημοσύνης από κειμενικές πηγές, δημιουργώντας ένα ενοποιημένο, επεξηγήσιμο αναλυτικό πλαίσιο. Η αρχιτεκτονική περιλαμβάνει: (i) χωροχρονική εναρμόνιση ετερογενών δεδομένων, (ii) εξαγωγή γεγονότων, οντοτήτων και τοποθεσιών ευθυγραμμισμένων με οντολογία, και (iii) αποθήκευση και συσχέτιση σε γράφο γνώσης Neo4j. Η μεθοδολογία GraphRAG (Graph-based Retrieval-Augmented Generation) επιτρέπει την παραγωγή τεκμηριωμένων, εξηγητικών αφηγήσεων που συνδέουν ανωμαλίες σε ενεργειακές χρονοσειρές με πιθανούς αιτιώδεις παράγοντες, διασφαλίζοντας ρητή ιχνηλασιμότητα και διαφάνεια.

Με τη γεφύρωση της στατιστικής ανίχνευσης ανωμαλιών με τη σημασιολογική κατανόηση, η εργασία συμβάλλει στην ανάπτυξη μιας Σημασιολογικής Χωρικής Υποδομής Δεδομένων (SSDI) για τον ενεργειακό τομέα, ενισχύοντας τη διαφάνεια, την ερμηνευσιμότητα και τη λήψη τεκμηριωμένων αποφάσεων στις αναλύσεις ενεργειακών δεδομένων.

Λέξεις-κλειδιά — ENTSO-E, ΑΔΜΗΕ, Γεωχωρικά Δεδομένα, Οντολογίες, Γράφος Γνώσης, LLMs, GraphRAG, LangChain, JSON Schema, Ανίχνευση Ανωμαλιών, Πρόβλεψη Ενέργειας.

Abstract

This thesis proposes an architectural framework for extracting semantic and spatial references from unstructured text using Artificial Intelligence, with application to the energy market domain. The European energy sector is rapidly digitalizing, producing vast volumes of open and semi-open data such as market indicators (ENTSO-E), transmission system measurements (ADMIE), geospatial datasets for infrastructures, and meteorological fields. Yet, structured data alone are rarely sufficient to explain market anomalies—sudden price spikes or shifts in production mix—whose causes are often described only in unstructured textual sources like news articles and policy announcements.

The proposed system integrates structured time-series with AI-driven knowledge extraction from textual data to form a unified, explainable analytical framework. Its architecture consists of: (i) spatiotemporal harmonization and canonicalization of heterogeneous datasets; (ii) ontology-aligned extraction of events, entities, and locations with semantic, spatial, and temporal grounding; and (iii) storage and reasoning within a Neo4j knowledge graph. The **GraphRAG** methodology (Graph-based Retrieval-Augmented Generation) enables provenance-rich, explainable narratives that link anomalies in energy time-series to their plausible drivers through Cypher-first retrieval strategies.

By bridging statistical anomaly detection with semantic understanding, the thesis contributes to the development of a **Semantic Spatial Data Infrastructure (SSDI)** for the energy sector. This framework enhances transparency, explainability, and decision support in energy data analytics, offering a foundation for future integration with predictive and causal models.

Keywords — ENTSO-E, ADMIE, Geospatial Data, Ontologies, Knowledge Graph, LLMs, GraphRAG, LangChain, JSON Schema, Anomaly Detection, Energy Forecasting.

Acknowledgements

I would like to express my deepest gratitude to my supervising professor, Vassilios Vescoukis, for his invaluable guidance, insight, and support throughout the course of this diploma thesis. His trust, patience, and scientific rigor were instrumental in shaping both the research direction and the final outcome of this work.

I also wish to warmly thank George Papakyriakopoulos, whose constructive feedback, and collaborative spirit greatly enriched this project. His experience in topology and semantic spatial data provided clarity and motivation during critical stages of development.

My sincere appreciation extends as well to all the professors of the School of Electrical and Computer Engineering at the National Technical University of Athens, for the knowledge, inspiration, and discipline they have imparted over the years of study.

I would also like to thank my colleagues and friends, whose discussions, companionship, and encouragement offered balance and perspective throughout this journey.

Above all, I am profoundly grateful to my family, for their unwavering support, patience, and sacrifices during all these years of study. Their faith in me, even in moments of doubt, made it possible to reach this point.

Contents

List of Figures

List of Tables

List of Listings

Abbreviation / Term	Short Definition
ENTSO-E	European Network of Transmission System Operators for
ΑΔΜΗΕ / ADMIE	Independent Power Transmission Operator of Greece (TSO)
MeteoSearch	Meteorological datasets and indicators for energy forecasting
Geospatial Data	Spatial/geographical data for infrastructures and administrative
CRS	Coordinate Reference System
EPSG:4326	WGS84 geodetic system (degrees)
GeoJSON	JSON format for geometries and attributes
Shapefile (SHP)	Widely used GIS file format (ESRI)
WKT/WKB	Well-Known Text / Binary — geometry representation formats
ISO 8601	International date/time format standard
Time Zone (IANA)	Time zone identifiers, e.g., *Europe/Athens*
PostgreSQL	Open-source relational database
PostGIS	PostgreSQL extension for GIS (spatial types and indexes)
QGIS	Open-source GIS software for mapping and spatial analysis
ETL	Extract-Transform-Load — data acquisition, transformation, and
Cypher	Query language for Neo4j property graphs
APOC	Neo4j procedures and functions library
Property Graph	Graph model with properties on nodes and relationships
Graph Traversal	Navigating a graph to retrieve or infer information
Ontology	Formal representation of concepts and relationships within a
Namespace Registry	Registry of namespaces/URIs (e.g., *ontology_registry.json*)
URI	Uniform Resource Identifier — unique resource reference
JSON Schema	Schema for validating JSON data structures
Semantic-fit	Degree of semantic compatibility between class/entity mappings
Bilingual Labels	English/Greek entity labels
Temporal Anchoring	Normalization of temporal references to instants or intervals
Provenance	Source traceability and processing lineage
LLM	Large Language Model

RAG	Retrieval-Augmented Generation — retrieval and context-
GraphRAG	RAG augmented with Knowledge Graph retrieval
KG	Knowledge Graph
KGQA	Knowledge Graph Question Answering
QA	Question Answering — automated query-response systems
LangChain	Framework for orchestrating LLMs and tool chains
GenAI Stack	Generative AI tool stack for intelligent applications
ChatGPT-4.1	General-purpose LLM used for QA and RAG
Ollama 2	Local LLM runtime/distribution tool
Prompt	Instructional text guiding LLM behavior
Few-shot	Prompting technique with a few in-context examples
CoT	Chain-of-Thought — guided reasoning approach
NER	Named Entity Recognition
RE	Relation Extraction
EE	Event Extraction
Coreference	Identification of identical entities in text
Semantic Fit Guard	Safeguard preventing invalid semantic matches
Temporal Commit Engine	Module for unifying and committing temporal references
Event Cascade Planner	Planner for associating and ordering event sequences
Patch Manager	Suggestion and compliance correction system
Theme Extractor	Thematic and topic extraction module
ARIMA/SARIMA	Classical statistical models for time series forecasting
LSTM	Long Short-Term Memory — recurrent neural network for time
AE/VAE	Autoencoder / Variational Autoencoder for anomaly detection
OCSVM	One-Class SVM — outlier detection algorithm
LOF	Local Outlier Factor — local anomaly score
Isolation Forest	Tree-based anomaly detection algorithm
GNN	Graph Neural Network
STGNN	Spatio-Temporal Graph Neural Network

Backtesting	Retrospective model testing on historical data
Cross-validation (CV)	Model validation via training/test splits
Rolling Window	Moving window for model training and evaluation
Feature Engineering	Construction of input features from raw data
Ablation	Controlled experiment by removing/adding components
MAPE	Mean Absolute Percentage Error
sMAPE	Symmetric Mean Absolute Percentage Error
MASE	Mean Absolute Scaled Error
RMSE	Root Mean Square Error
MAE	Mean Absolute Error
MSE	Mean Square Error
Directional Accuracy (DA)	Percentage of correct direction predictions
Precision/Recall/F1	Extraction and classification performance metrics

Greek Extended Abstract

Η παρούσα διπλωματική εργασία προτείνει και υλοποιεί ένα αρχιτεκτονικό πλαίσιο για την εξαγωγή σημασιολογικών και χωρικών αναφορών από αδόμητο κείμενο με χρήση Τεχνητής Νοημοσύνης, με εφαρμογή στον τομέα των ενεργειακών αγορών. Ο ευρωπαϊκός ενεργειακός τομέας ψηφιοποιείται ραγδαία και παράγει μεγάλα σύνολα ανοικτών και ημι-ανοικτών δεδομένων: δείκτες και τιμές αγοράς από την πλατφόρμα ENTSO-E, μετρήσεις συστημάτων μεταφοράς από φορείς όπως ο ΑΔΜΗΕ, γεωχωρικά επίπεδα για υποδομές, ζώνες προσφορών και ελέγχου, διοικητικές ενότητες, καθώς και μετεωρολογικά πεδία. Τα δεδομένα αυτά είναι δομημένα, μηχανικά επεξεργάσιμα και συνοδεύονται από τεκμηριωμένα μεταδεδομένα. Παρ' όλα αυτά, τα ποσοτικά σήματα από μόνα τους σπάνια επαρκούν για να εξηγήσουν ανωμαλίες, όπως αιφνίδιες «εκρήξεις» τιμών ή μεταβολές στο μίγμα παραγωγής, των οποίων τα αίτια αποτυπώνονται κυρίως σε αδόμητες πηγές κειμένου (ειδησεογραφία, ανακοινώσεις πολιτικής, τεχνικές αναφορές).

Αφετηρία της εργασίας αποτελεί το κενό μεταξύ αυτών των δύο κόσμων: των χρονοσειρών με σαφή φυσική και οικονομική σημασία και των ετερόκλητων, κειμενικών πηγών που περιγράφουν γεγονότα, φορείς, αποφάσεις και περιστατικά. Στόχος είναι η ανάπτυξη μιας αρχιτεκτονικής που συνδυάζει παραδοσιακή διαχείριση δομημένων δεδομένων με λειτουργίες υποστηριζόμενες από σύγχρονα Μεγάλα Γλωσσικά Μοντέλα (LLMs), ώστε να καταστεί δυνατή η αυτόματη σύνδεση «ποσοτικών» ανωμαλιών με «ποιοτικά» εξηγητικά γεγονότα.

Το προτεινόμενο σύστημα ακολουθεί μια πολυστρωματική αρχιτεκτονική:
(α) Χωροχρονική εναρμόνιση ετερογενών πηγών δεδομένων, με κανονικοποίηση χρονοσήμανσης σε UTC, ενοποίηση χωρικών αναφορών σε κοινό σύστημα αναφοράς (CRS) και χρήση κανονικοποιημένων ταυτοτήτων για ζώνες και χώρες.
(β) Εξαγωγή γνώσης από κείμενα, μέσω ενός αγωγού (pipeline) εξαγωγής γεγονότων, οντοτήτων, θεματικών και τοποθεσιών, ευθυγραμμισμένων με οντολογία. Η εξαγωγή βασίζεται σε LLMs με ontology-aware prompting και schema-constrained JSON έξοδο, ώστε τα αποτελέσματα να είναι δομικά έγκυρα, χρονικά αγκυρωμένα (ISO 8601) και χωρικά γειωμένα.
(γ) Αποθήκευση σε γράφο γνώσης Neo4j, όπου χρονοσειρές, ανωμαλίες, γεγονότα, οντότητες, θέματα και ζώνες αναπαρίστανται ως κόμβοι και σχέσεις σε property graph με περιορισμούς

και δείκτες. Πάνω σε αυτόν τον γράφο υλοποιείται GraphRAG (Graph-based Retrieval-Augmented Generation), το οποίο επιτρέπει την ανάκτηση τεκμηριωμένων υπογραφημάτων γύρω από χρονικά παράθυρα ανωμαλιών και τη σύνθεση επεξηγηματικών αφηγήσεων.

Στο επίπεδο εφαρμογής, η εργασία υλοποιεί ένα πλήρες pipeline: ανίχνευση ανωμαλιών σε χρονοσειρές ENTSO-E με χρήση ανθεκτικού z-score, εισαγωγή των ανωμαλιών και των χρονοσειρών στον γράφο γνώσης, εξαγωγή και ευθυγράμμιση γεγονότων από ένα επιλεγμένο σώμα περίπου 250 ελληνικών άρθρων ενέργειας, καθώς και ανάπτυξη διαδραστικού χρονοδιαγράμματος (Flask + Highcharts) που επιτρέπει σε αναλυτές να επιλέγουν ανωμαλίες, να βλέπουν σχετιζόμενα γεγονότα, οντότητες και θέματα, και, εφόσον ενεργοποιηθεί, να λαμβάνουν GraphRAG-βασισμένες, τεκμηριωμένες εξηγήσεις.

Ως μελέτη περίπτωσης εξετάζεται ανωμαλία σε τιμές day-ahead στην ελληνική αγορά, σε χρονικό διάστημα όπου συμπίπτουν γεωπολιτικές εντάσεις και μεταβολές στο μίγμα παραγωγής. Η ανωμαλία λειτουργεί ως «άγκυρα» για την ανάκτηση σχετικών γεγονότων (π.χ. ενεργειακή κρίση, ζητήματα τροφοδοσίας φυσικού αερίου, θερμικά κύματα), ενώ η παραγόμενη αφήγηση παραμένει πλήρως ιχνηλάσιμη, καθώς κάθε ισχυρισμός συνδέεται με συγκεκριμένους κόμβους και πηγές στον γράφο γνώσης.

Τέλος, η εργασία συζητά τους περιορισμούς της προσέγγισης (κάλυψη δεδομένων, εξάρτηση από LLMs, υπολογιστικό κόστος) και χαράζει μελλοντικές κατευθύνσεις, όπως πολυγλωσσική επέκταση, ενσωμάτωση αιτιοκρατικά ευαισθητοποιημένης ανάκτησης και κλιμάκωση της αρχιτεκτονικής σε παραγωγικά περιβάλλοντα. Συνολικά, η συμβολή της διατριβής εστιάζει στη γεφύρωση στατιστικής ανάλυσης χρονοσειρών και σημασιολογικής κατανόησης, ως βήμα προς μια ώριμη Σημασιολογική Χωρική Υποδομή Δεδομένων για τον ενεργειακό τομέα.

CHAPTER 1

1. INTRODUCTION

1.1. Motivation and Context

The motivation for this work comes from the European power sector, which is rapidly digitizing. Open and semi-open datasets are now common, to support transparency in markets operation: market prices and indicators (ENTSO-E), transmission-system measurements provided by operators such as ADMIE, geospatial references to infrastructures, biding zones, control zones, administrative areas, meteorological data, and more. Such datasets come in structured machine-processable formats, and are supported by well-documented metadata. However, examining structured data alone is insufficient to explain anomalies - that is, abrupt changes or unexpected patterns such as the sudden surge in energy prices following an external event, or the shifts in generation mix resulting the introduction of environmental policies. Understanding such phenomena requires a broader layer of expert knowledge, much of which is disseminated in textual form through news articles, policy reports and official announcements.

The motivation of this work comes from the challenge of developing software with architectures that seamlessly integrate traditional data management with AI-driven operations. The goal is to unify heterogenous data sources - ranging from structured datasets to unstructured textual mentions in news feeds—within a common analytical framework. Such integration enables applications that combine all relevant technical, economic, and contextual information in both structured and unstructured formats, supporting situational awareness and decision-making.

In the broader context, this effort contributes to the development of a Semantic Spatial Data Infrastructure (SSDI) for the energy sector. The envisioned SSDI aims to facilitate applications that assist both experts and non-experts in interpreting and understanding non-typical behaviors observed in quantitative energy data. It does so by leveraging spatially referenced and semantically aligned information extracted from unstructured sources, thus bridging the gap between data-driven analysis and knowledge-driven interpretation.

Needless to say, the ecosystem encompassing all relevant energy data is inherently heterogeneous and fragmented. Data are collected and published in diverse formats (e.g., CSV, JSON, APIs), often employing inconsistent terminology to describe the same concepts (for instance, region, bidding zone, or περιφέρεια). They also appear in multiple languages, coordinate reference systems, and temporal resolutions (such as 15-minute, hourly, or daily intervals). Every energy-related signal is intrinsically spatiotemporal—it holds meaning only when its spatial context (node, region, or country) and temporal context (timestamp and time zone) are clearly defined, together with associated factors such as weather, grid state, policy environment, and geopolitical events. In the absence of consistent spatiotemporal harmonization and auditable provenance, both knowledge integration and explainability are fundamentally undermined.

1.2. Problem Statement and Gap

European power market timeseries often show sharp fluctuations—price spikes, demand drops, or generation anomalies—that are hard to explain post hoc. Structured time-series and unstructured news are usually processed in separate pipelines; especially unstructured news are only "processed" by human experts. As a result, the automatic detection of semantic links between measured anomalies and qualitative relevant descriptions is left to human experts that need to manually operate the collection, interpretation and linking of relevant data. Therefore, a gap is recognized: is it possible to use modern AI tools to link quantitative structured data with well-defined semantics to semantically aligned data from free-text public sources, such as newsfeeds? To achieve this, we would need an application architecture that enables:

- normalizing spatial and temporal references across heterogeneous sources

- Extracting and mapping to relevant ontologies data from text with proper semantic, spatial and temporal references (eg ISO 8601 for time anchoring, geospatial grounding)

- fuses everything to be represented in a queryable knowledge graph that supports retrieval-augmented explanations with explicit source attribution.

1.3. Research questions

This thesis builds an end-to-end system that couples explicit time-series with implicit knowledge from free-text articles to explain anomalies and support domain analyses of all kinds.

- RQ1 (Data). How can heterogeneous datasets (ENTSO-E, ADMIE, geospatial, meteorological) be harmonized in space and time (UTC, CRS, granularity) with reproducible transformations and evidence-preserving exports?

- RQ2 (Ontology extraction). Can an ontology-aligned pipeline reliably detect events, entities, locations, and intervals from Greek energy news (with English extension in scope) and map them to a controlled standard vocabulary with schema-level guarantees?

- RQ3 (Graph reasoning). How can a knowledge graph with canonical identifiers, constraints, and time indexing, enable effective GraphRAG retrieval that links anomalies to plausible drivers via Cypher-first strategies?

- RQ4 (Explainability and prediction support). To what extent can anomaly explanations and extracted signals support informed energy market decisions? (Quantitative forecast integration is flagged for future work; current focus is on explanation quality and data readiness.)

1.4. System Architecture and Core Contributions

This section summarizes the principal components and technical contributions of the developed system. Each component corresponds to a distinct layer of the end-to-end architecture, designed to integrate heterogeneous energy datasets and unstructured text sources into a unified, semantically enriched analytical framework. The emphasis lies on modularity, reproducibility, and explainability—ensuring that every transformation, extraction, and inference step remains both transparent and verifiable.

• **Layered reference architecture.** Clear separation of acquisition, staging and analytics (InfluxDB), semantic extraction (ADK agents), graph persistence (Neo4j), and application layers. Implemented through seven refined notebooks and a production-ready command-line interface (ProductionApp).

• **Spatiotemporal harmonization and canonicalization.** UTC-normalized timelines, CRS-aware geospatial handling, and a PostGIS-backed location canonicalizer that eliminates country and location duplicates, enforces ISO-aligned identifiers, and ensures idempotent Cypher upserts under strict schema constraints and indexes.

• **Article extraction with guarantees.** ADK-based agent producing ontology-aligned JSON with ISO 8601 anchoring; deterministic fallback dataset for offline, reproducible runs; validated on a curated corpus of 250 Greek energy articles with direct extensibility to larger multilingual collections.

• **Knowledge graph and GraphRAG.** Canonicalized spatial and semantic entities, time-tree indexing, schema constraints, and Cypher-first retrieval for anomaly context assembly; integrated with notebook-driven RAG explainability and a real-time anomaly timeline service.

• **Explainability interfaces.** Retrieval-augmented narratives linking ENTSO-E anomalies to article-derived events; interactive timeline built with Flask and Highcharts featuring multi-parameter filtering (temporal, geographic, and ontological) and structured prompt generation with explicit citation control.

• **Provenance, validation, and reproducibility.** Comprehensive evidence tagging, dataset and version logging, exportable CSV/Parquet artifacts, offline-safe execution modes, and validation scripts for ingestion, persistence, and visualization components.

1.5. Thesis Structure

• **Chapter 2 – Background and Theoretical Context.** Reviews the foundational concepts underlying the work, including energy time series, anomaly detection methodologies, ontologies, knowledge graphs, GraphRAG frameworks, and agentic workflows.

• **Chapter 3 – System Architecture.** Describes the overall architectural framework, including the design drivers, layered structure, data contracts, and cross-cutting components integrating Neo4j, InfluxDB, PostGIS, and ADK-based semantic extraction modules.

• **Chapter 4 – Implementation of the Processing Pipeline.** Presents the software realization of the end-to-end workflow, detailing the design, data ingestion mechanisms, transformation logic, and orchestration of analytics across layers.

• **Chapter 5 – Case Study and Explainability Demonstration.** Examines a representative anomaly in Greek day-ahead electricity prices, showcasing how GraphRAG-based retrieval and the interactive timeline service enable transparent, evidence-grounded explanations.

• **Chapter 6 – Conclusions and Future Research.** Synthesizes the findings of the study, outlines current limitations, and proposes directions for scaling, optimization, and continued integration of AI-driven methods into energy analytics.

Use of Artificial Intelligence

Artificial Intelligence (AI) tools were employed as supportive instruments during the preparation of this thesis to enhance clarity, productivity, and technical precision. AI tools supported parts of this work by proposing code snippets, drafting brief text, and sketching diagrams. The author verified, adapted, and finalized all content.

Large Language Models (LLMs), including, were used under supervision to assist in section structuring and diagram generation. In particular, LLMs contributed to the production of PlantUML-based system architecture diagrams and Graphviz (DOT-language) knowledge-graph visualizations, which were subsequently reviewed and edited by the author for accuracy and consistency.

The role of these tools was strictly auxiliary—serving as aids rather than substitutes for human reasoning or original authorship. All conceptual design, analytical interpretation, and final synthesis were independently conducted by the author. The integration of AI contributed to improved organization and visualization of complex methodological ideas, resulting in a coherent, reproducible, and well-documented research framework that fully respects academic integrity.

CHAPTER 2

2. BACKGROUND & REQUIREMENTS

2.1. Energy Market Data

The system builds upon heterogeneous open datasets that provide explicit spatial and temporal references. At its core lies the ENTSO-E Transparency Platform, which offers pan-European electricity time series through a RESTful API (XML/CSV, API-key authentication). In this work, the following ENTSO-E series are utilized: Day-Ahead Price (€/MWh), Actual Total Load (MW), and Generation per Type (MWh) for the Greek bidding zone and, where available, selected neighboring zones.

All series are harmonized to Coordinated Universal Time (UTC) at hourly granularity, following consistent aggregation rules per metric (e.g., mean for MW or price, sum for MWh). Anomalies in these reference timelines are detected using a robust z-score method applied over rolling weekly windows, employing a median/MAD estimator with thresholds configurable within the analytical notebooks. Each detected anomaly is materialized as an Anomaly entity and persisted in the knowledge graph, ensuring that the analytical scope (series and granularity) and the applied method (detector) remain explicit and fully reproducible.

To complement the structured series, a curated corpus of approximately 250 Greek-language energy-news articles has been integrated from specialized portals. These are processed through a bilingual, ontology-aligned extraction pipeline that performs ISO 8601-compliant temporal anchoring and geospatial reference mapping. The pipeline identifies events, entities, and relationships, which are subsequently persisted as interconnected nodes and edges within the knowledge graph. Each extracted payload carries evidence tags for source provenance, ensuring traceability. These textual extractions are leveraged by GraphRAG to contextualize and explain

anomalies—or, more generally, to elucidate non-typical behaviors observed in energy-market time series such as those obtained from ENTSO-E.

A complementary structured source is ADMIE (IPTO), the Greek Transmission System Operator. ADMIE publishes both network topology data—including substation coordinates, 400 / 150 kV transmission lines, and cross-border interconnections—and operational or market information. Within this work, emphasis is placed on representing the physical grid topology, which is not covered by ENTSO-E. Network geometries are persisted in standard geospatial formats to support efficient spatial queries and to map physical assets to knowledge-graph elements, such as GridNode entities and Line relationships with associated geometries. Representing the approximately 11,800 km high-voltage network as spatially referenced graph structures enables direct correlation between detected events (e.g., outages, maintenance operations) and their physical grid locations.

Finally, meteorological data are incorporated via MeteoSearch (National Observatory of Athens), which provides historical measurements of temperature, humidity, wind, and precipitation per station. These series are aligned to hourly UTC, using consistent aggregation rules (e.g., hourly mean temperature or hourly total precipitation), normalized for units and data quality, and spatially linked to administrative regions through open geospatial boundaries in PostgreSQL / PostGIS. This integration enables composite queries such as *"What were the demand and temperature in Attica on date D?"* and supports proximity or in-region event analysis—thereby enriching structured market data with environmental and spatial context.

2.2. Forecasting & Anomaly Detection

6.3.1. Energy Time-Series Characteristics

Energy-system signals display complex and distinctive behaviors that influence both forecasting methodologies and anomaly-detection approaches. Their variability arises from the interaction of physical, socio-economic, and regulatory processes, resulting in strong and overlapping seasonalities. Physical cycles such as solar irradiance and temperature interact with socio-economic patterns (workdays, weekends, and holidays) and market procedures (e.g.,

gate-closure times and settlement windows). Consequently, daily and weekly periodicities often coexist with annual components, while transitions such as daylight-saving adjustments introduce discontinuities and subtle calendar drift.

A further defining trait of these series is non-stationarity. Long-term transformations—such as increased renewable penetration, regulatory reforms, or infrastructure changes (e.g., new interconnectors, plant outages)—frequently alter their statistical properties, including level, variance, and frequency composition.

Volatility and heavy-tailed distributions are particularly prominent in electricity price series, where supply–demand imbalances, fuel-cost dynamics, and scarcity pricing mechanisms generate sharp spikes and outliers. Many of these dynamics are driven by exogenous factors—including weather conditions, fuel markets, policy interventions, and planned or forced outages—making purely endogenous models insufficient to capture the full causal structure unless they incorporate external explanatory variables.

These properties underscore the need for robust preprocessing pipelines and anomaly detectors capable of maintaining reliability under regime shifts, non-stationary noise, and heterogeneous variance patterns.

6.3.2. Anomaly Taxonomy

Anomalies in energy time series can be classified according to their temporal structure and contextual dependencies:

- **Point anomalies:** isolated spikes or drops that deviate sharply from the local statistical distribution.
- **Contextual anomalies:** values that are atypical given the prevailing conditions (for example, unusually low weekday demand during cold weather), even if they are not globally extreme.
- **Collective anomalies:** sequences of observations—such as sustained runs, plateaus, or oscillatory patterns—that appear anomalous only when considered together rather than individually.

- **Regime changes:** structural shifts in level, variance, or trend, such as persistent price elevations following policy interventions or market design changes, often modeled as change points.

In the energy domain, contextual interpretation is essential. A sharp price surge occurring alongside interconnector maintenance or extreme heat may be entirely plausible when the underlying conditions are known, whereas the same event would seem anomalous in isolation.

Therefore, effective frameworks must distinguish between statistical detection, which identifies candidate anomalies, and contextual reasoning, which evaluates their plausibility and causal explanations.

6.3.3. Forecasting Families

Forecasting methodologies in the energy domain span statistical, machine learning, deep learning, and spatio-temporal paradigms, each offering distinct advantages and trade-offs depending on data characteristics and operational objectives.

Statistical models such as ARIMA/SARIMA and exponential smoothing (Holt–Winters) remain foundational for modeling stationary or quasi-stationary seasonal components. State-space formulations and extensions like TBATS can accommodate multiple or long seasonalities and complex calendar effects. Their strengths lie in parsimony, interpretability, and robust uncertainty quantification, provided that the underlying assumptions of limited non-stationarity hold.

Machine learning methods, including gradient boosting and other tree-based ensembles, capture nonlinear dependencies among engineered features such as lagged observations, rolling statistics, calendar indicators, and exogenous variables (e.g., temperature or fuel prices). These approaches often improve predictive accuracy under nonlinear dynamics but depend critically on careful feature design and continuous monitoring to mitigate concept drift.

Deep learning models—notably recurrent neural networks (RNNs/LSTMs), temporal convolutional networks (TCNs), and Transformer-based architectures—excel at modeling long-range dependencies and complex temporal interactions, particularly when large datasets with rich covariates are available. While these architectures can process raw or lightly engineered inputs, they typically trade interpretability for capacity, requiring extensive regularization, hyperparameter tuning, and validation to avoid overfitting.

Spatio-temporal approaches explicitly incorporate the network structure of the energy system—zones, substations, and interconnections. Graph neural networks (GNNs) propagate information along the grid topology, capturing spatial correlations and constraints. These methods are particularly effective when system-wide interactions significantly influence local behavior, such as cross-border energy flows or regional weather dependencies.

Finally, hybrid strategies combine multiple paradigms. Examples include forecast-then-residual pipelines, where statistical or machine-learning models capture expected behavior and anomaly detection operates on residuals, or ensemble frameworks that integrate heterogeneous predictors to enhance stability and accuracy across varying market regimes[11],[12],[13].

6.3.4. Anomaly Detection Methods

Within electricity markets and power-system operation, anomaly detection forms a cornerstone of situational awareness. Analysts and operators are not concerned with every minor deviation from normal behavior, but rather with salient departures—events such as extreme price spikes, abrupt demand drops, or structural changes in the generation mix—that warrant interpretation.

In this thesis, anomalies are not treated as endpoints but as anchors for downstream explainability: once an anomaly window is identified, the system queries unstructured sources—such as news, policy statements, or operational bulletins—to uncover plausible explanatory events.

Understanding the principal families of anomaly detection methods is therefore essential for motivating the simple, robust detector adopted in this work.

CHAPTER 2: BACKGROUND & REQUIREMENTS

Families of Anomaly Detection Methods

Detection methods differ primarily in how they model "normal" behavior and what statistical or structural assumptions they impose.

Distribution-based (univariate) detectors identify anomalies as standardized deviations from a reference distribution. Classical z-score methods assume stable mean and variance, while robust variants replace these with the median and median absolute deviation (MAD) or interquartile ranges, reducing sensitivity to outliers and heavy tails (e.g., Hampel filters).

Seasonal hybrids remove recurring cycles through decomposition (e.g., STL or LOESS) and detect outliers on deseasonalized residuals, improving precision for strongly periodic signals such as load or generation.

Residual-based detectors operate on the difference between observed and forecasted values derived from statistical, machine-learning, or deep-learning models. Control-chart families—including Shewhart, EWMA, and CUSUM—evaluate deviations, persistent shifts, or trends in residuals, offering clear probabilistic thresholds under known distributional assumptions.

Change-point detection targets structural breaks in level or variance. Methods like PELT (Pruned Exact Linear Time) efficiently identify multiple change points, while Bayesian online change-point detection provides a probabilistic, sequential treatment well-suited to streaming data. These methods capture regime shifts effectively but may overlook short-lived spikes.

Density- and isolation-based approaches—such as Local Outlier Factor (LOF), DBSCAN, or Isolation Forest—define dense "normal" regions in feature space and flag low-density or easily isolated points as anomalies. Although flexible and capable of detecting nonlinear structures, they require careful feature design and are less interpretable for strictly time-ordered data.

Multivariate and probabilistic models exploit dependencies across multiple series. Techniques like Robust PCA or subspace tracking detect deviations from low-rank structure, while multivariate state-space models identify anomalies through breakdown of correlations among variables.

Probabilistic forecasting frameworks further generalize this by estimating predictive intervals, treating anomalies as low-probability observations conditioned on known covariates.

Graph-based anomaly detection extends these concepts to networked systems. It identifies deviations in a node's behavior relative to its neighbors or unexpected disruptions in flow patterns across edges. Temporal graph extensions capture evolving topology and attributes, making them particularly suitable for power-grid infrastructures.

Method Selection

No single technique performs optimally across all regimes.

- Robust univariate methods offer transparency, computational efficiency, and reproducibility.
- Residual and change-point approaches capture dynamic shifts when dependable baselines exist.
- Multivariate and graph-based models reveal cross-series or topological dependencies but require richer data and introduce higher complexity.

For the purposes of this thesis, a robust z-score detector (MAD-based) was chosen. It balances simplicity, interpretability, and stability, ensuring reproducible anomaly flags suitable for linking to contextual explanations via the knowledge graph.

Robust MAD-based z-score as the working detector

Within this spectrum, the proposed system adopts a robust MAD-based z-score as its primary anomaly detector for structured time series (e.g. prices, loads, generation by fuel). The rationale is twofold. First, the method is transparent and easily interpretable: anomalies correspond to observations that lie several robust standard deviations away from a typical level. Second, it is lightweight and reproducible, making it suitable as a stable anchor for downstream explainability and case studies, with more sophisticated methods left as future extensions.

Statistic

Given a time series y:

- Compute the median m and the Median Absolute Deviation (MAD):

  $MAD = median(|y - m|)$

- Define the standardized robust z-score:

  $z = 0.6745 \times \frac{(y-m)}{MAD}$ (The constant 0.6745 scales MAD to the standard deviation σ under normality.)

Thresholding and Level Shifts

- Flag an anomaly when $|z| \geq \tau$.
- Detect a level shift when $|y_t - y_{t-1}| \geq \phi \times max(|y_{t-1}|, \varepsilon)$, labeled as **LevelShift::Rise** or **LevelShift::Drop** to distinguish persistent step changes from isolated spikes.

Parameter Choices

- $\tau = 3.5$: default robust threshold.
- $\tau = 2.5$: for energy prices, capturing known spikes without over-alerting.
- $\phi = 0.15$: detects regime-like step changes.
- Minimum points = 8: ensures stability over short segments.

Listing 6.3.4-1: MAD_based z-score

Greece Example (Weekly, 2021-01-03 → 2022-12-25)

- Day-ahead prices peak at 604.14 €/MWh on 2022-08-28 with z = 3.06 (LevelShift::Rise).
- Lignite share (lignite generation ÷ total load) jumps to 21.23 % on 2022-08-21 (z = 2.95) and 24.94 % on 2022-12-11 (z = 4.00), both LevelShift::Rise.
- Yearly context: mean price rises from 120 €/MWh (2021) to 289 €/MWh (2022); lignite share mean from 10.5 % to 11.4 % with higher extremes.

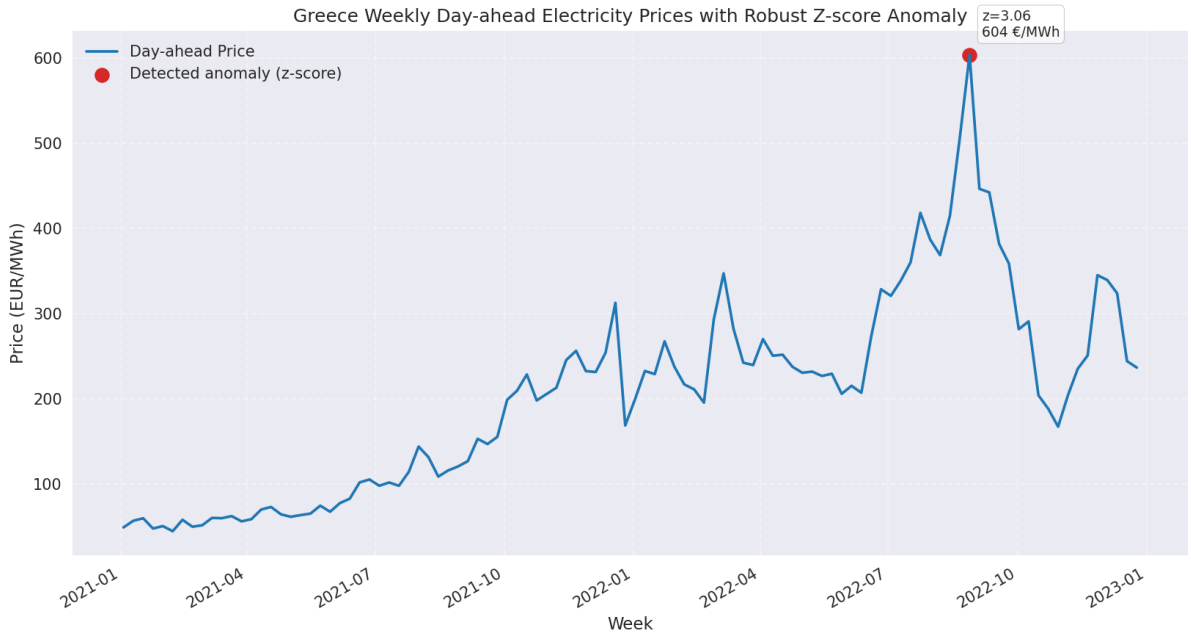


Figure 2-2: Greece Weekly Day-ahead Electricity Prices with Robust Z-score Anomaly

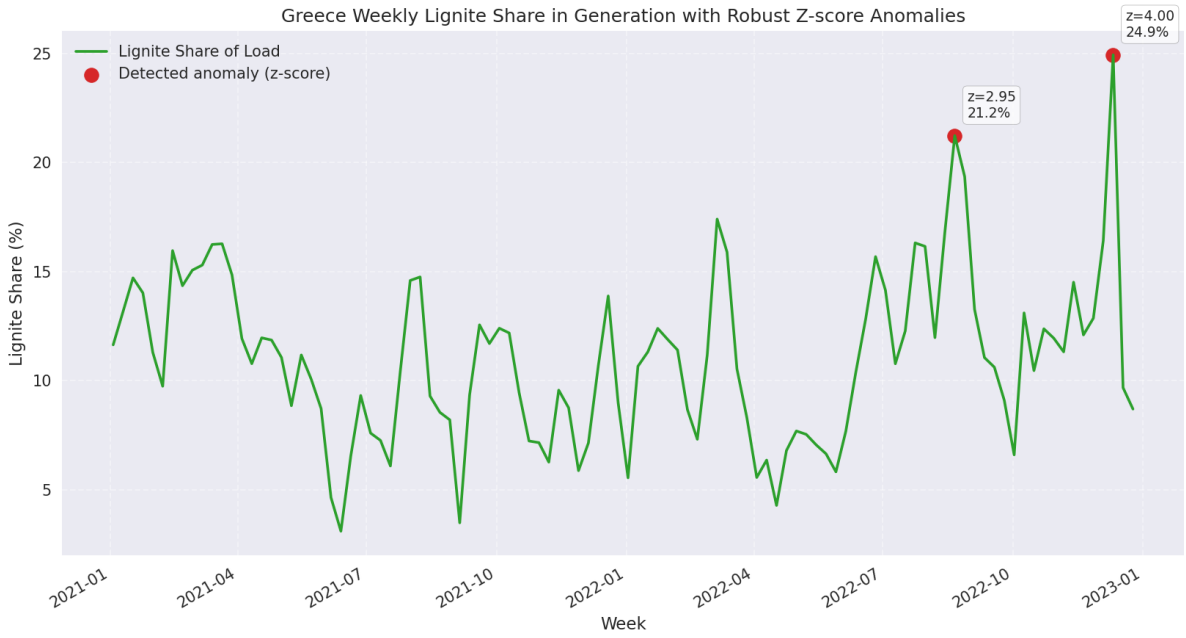


Figure 2-1: Greece Weekly Lignite Share in Generation with Robust Z-score Anomalies

These spikes align with the 2022 Russian invasion of Ukraine. Constrained gas flows drove wholesale electricity prices to records by late summer (the price anomaly), while Greece leaned more on domestic lignite through H2 2022 (the share anomalies).

6.3.5. Explainability and Contextualisation

Statistical flags are necessary but insufficient for decision support. Anomalies become meaningful when linked to plausible drivers—policy changes, outages, weather extremes, or market dynamics—grounded in space and time. Knowledge graphs provide a structured representation of entities, events, and relationships, enabling multi-hop reasoning and explicit provenance. Retrieval-augmented generation (RAG) layered over a graph can assemble contextual evidence around an anomaly window and synthesize coherent narratives, while reducing hallucinations relative to ungrounded text generation.

Causality-aware retrieval strengthens explanations by enforcing temporal precedence and plausible pathways (e.g., event → constraint → price effect), though formal causal discovery remains challenging in practice. Human-in-the-loop review—supported by citations, graph paths, and uncertainty cues—helps calibrate trust and refine rules iteratively.

6.4. LLMs for Knowledge Extraction

The rise of large language models (LLMs) has fundamentally changed how knowledge can be extracted from unstructured text. Unlike traditional NLP pipelines that rely on fixed pattern-matching or manually engineered rules, LLMs provide adaptive, context-aware reasoning that can interpret diverse textual expressions of the same underlying concept. In the energy sector, where valuable information is dispersed across technical reports, regulatory announcements, and journalistic narratives, this capability enables the automatic identification of relevant events and entities — such as outages, market interventions, and extreme weather — directly from text.

LLM-based extraction acts as the bridge between unstructured human language and structured representations used in databases or knowledge graphs. Through prompting, fine-tuning, or schema-constrained generation, models can produce machine-readable outputs (e.g., JSON) that map linguistic information to defined ontological classes, properties, and relationships. This approach allows analysts to integrate news and policy documents seamlessly with time-series data, improving both explainability and situational awareness.

2.3.1. What is an Ontology?

An ontology formally defines the concepts, relationships, and attributes within a domain of knowledge. In practice, it serves as a shared vocabulary that enables both humans and machines to reason about data in a consistent, semantically meaningful way. Ontologies differ from simple taxonomies by encoding not only hierarchical relationships ("is-a") but also complex interdependencies such as "causes," "affects," "belongs to zone," or "produced by."

In the context of energy informatics, an ontology may include entities like PowerPlant, BiddingZone, WeatherEvent, and PolicyIntervention, and relations such as locatedIn, causes, or correlatesWith. When text-extracted information is aligned with these ontological structures, it becomes interoperable with other datasets, supports logical reasoning, and facilitates cross-source integration within a knowledge graph.

2.3.2. Motivation for Ontology-Aligned Extraction in Energy News

In the energy domain, critical facts and events often appear in unstructured text such as news articles, press releases, and reports. For example, a Greek news article might describe a power outage, a policy change, or an extreme weather event affecting the energy grid. Transforming these unstructured narratives into ontology-aligned facts is essential for downstream reasoning: a knowledge graph of energy events and entities can be queried and analyzed alongside time series data. The motivation is to bridge the gap between textual information and data-driven analysis – turning free text into structured knowledge that conforms to a domain schema. By aligning extracted information to a predefined ontology of energy concepts (e.g. classes like Infrastructure, PowerPlant, BlackoutEvent, and relations like LOCATED_IN or HAS_CAUSE), we ensure semantic consistency and make the information machine-interpretable. This approach enables the system to incorporate real-world context (events, actors, conditions) into tasks such as anomaly explanation, without manual data entry. In short, ontology-aligned knowledge extraction allows unstructured energy news (in Greek or other languages) to be converted into a graph of facts – who, what, when, where – that can enrich analytical models with real-world context.

From a theoretical standpoint, this falls under information extraction (IE), where the goal is to identify entities, relationships, and events in text[1]. Traditional IE pipelines use task-specific models (for example, separate named entity recognition and relation classification modules), which require substantial training data for each label type. In contrast, recent Large Language Models (LLMs) provide a more unified approach: a single generative model can perform joint extraction of entities and relations by parsing text and producing a structured output in one go. In the energy news context, a generative LLM can be prompted to read an article about, say, an unexpected power demand spike and output a JSON record of the salient TemporalEvent (e.g. an extreme heatwave) with its date and location. The unified approach is appealing for low-resource settings like Greek-language energy news, where training data for specialized IE models is scarce. By leveraging a powerful multilingual LLM with zero-shot or few-shot prompting, we can extract the needed information without task-specific fine-tuning. This generative IE paradigm has gained traction as it simplifies the pipeline – one model and prompt can handle multiple IE subtasks simultaneously. It is especially useful when the ontology of interest is custom (here, tailored to energy domain concepts) and one cannot easily find off-the-shelf models for each class.

However, using LLMs for knowledge extraction also introduces challenges. Generative models may hallucinate facts or produce text not faithful to the source. They might also miss details or output information in an inconsistent format. Thus, the pipeline must be designed with strict controls to ensure reliability and precision despite the flexibility of LLMs. In the following subsections, we discuss theoretical techniques to guide LLM outputs: ontology-aware prompting, schema-constrained generation, and normalization of temporal and geospatial references. Together, these ensure that the extracted knowledge is semantically accurate, structured, and grounded in the source text, which is crucial for building a trustworthy energy knowledge base.

2.3.3. Generative Information Extraction with LLMs

Large Language Models can perform Generative Information Extraction (GIE), meaning they generate structured outputs (like JSON or triples) that represent information in text[1]. Unlike pipeline IE (with separate NER, relation extraction, event detection stages), an LLM can take a passage and produce an all-in-one annotated representation. For example, given a news snippet

“The Greek power grid operator announced rolling blackouts in Attica on July 24 due to a heatwave”, a single prompt to the LLM can yield a structured record of a BlackoutEvent with properties: date = 2025-07-24, location = "Attica, Greece", cause = "heatwave", involved_actor = "Greek power grid operator", etc. This unified generative approach treats IE as a conditional text generation task, where the model outputs a representation of the salient facts.

The advantage of GIE is its flexibility: the same model can be instructed on different schemas or ontologies without retraining, simply by changing the prompt. This is particularly useful in our case, where the ontology is custom and evolving. Studies have shown that large LLMs are capable of parsing and transforming text into structured formats in few-shot settings. Furthermore, the model can inherently do zero-shot generalization to entity names or event types it hasn’t seen in training, as long as the prompt provides sufficient context. This is valuable for energy news, where new names (e.g. a newly commissioned power plant) or emerging event types (a novel market policy) might appear[3].

On the other hand, there are trade-offs versus traditional task-specific models. A fine-tuned NER model or relation classifier might achieve higher precision on a narrowly defined task, especially if plenty of labeled data exists. LLMs operating via prompts may sometimes omit required details or introduce errors if the prompt is not precise. Generative IE can struggle with over-generation (including irrelevant info) or format errors if not properly constrained. Additionally, LLMs have a tendency to produce answers regardless of confidence, which can lead to spurious extractions if the prompt does not enforce caution. Despite these challenges, the generative approach is appealing for our scenario because of its speed of development (no need to train multiple models) and its ability to capture complex relations in a single shot. With careful prompt engineering and constraints, we can mitigate many of the risks and obtain high-quality structured data from the LLM.

Recent surveys (e.g. Xu et al., 2024) conclude that LLM-based IE is a promising direction, especially when dealing with complex or low-resource domains[3]. By leveraging the world knowledge and language understanding of LLMs, we can extract facts that more brittle, small models might miss. In summary, our approach employs a generative LLM to perform end-to-end extraction of energy domain knowledge. The subsequent sections outline how we constrain and guide this generative process – using an ontology schema, structured output format, and normalization – to ensure the results are accurate and ready to integrate into a knowledge graph.

2.3.4. Ontology-Aware Prompting and Schema Constraints

A key design principle is to make the LLM aware of the target ontology and required output structure. Ontology-aware prompting means we explicitly inform the model about the classes and relations it should use, effectively whitelisting the allowed schema. In practice, the prompt includes a brief excerpt or description of the ontology schema relevant to extraction. For instance, we might list the main entity types (e.g. PowerPlant, GovernmentAgency, WeatherEvent, OutageEvent, PriceSpike) and their properties or relationships. By providing this context, the model's output space is narrowed to our domain vocabulary – it knows to prefer terms like "PowerOutage" or "Heatwave" if those are in the schema, rather than inventing unrelated categories. This technique leverages the LLM's ability to follow instructions: we give it a semantic guideline so that extracted facts are semantically aligned with our knowledge graph design. Empirically, ontology-grounded prompting reduces "drift" where the model might otherwise output information that doesn't fit our schema (e.g. an irrelevant relation or an undefined class). It also improves consistency, as the same concept will be referred to by the canonical name defined in the ontology (e.g. always output "region": "West Macedonia" instead of sometimes saying "Western Greece" for the same region).

Raw Term(s) / Phrase(s)	Canonical Term	Notes
“day-ahead price”, “DAM price”, “DA MCP”, “market clearing price”	`metric_type=day_ahead_price`	Normalize MCP/MWh context
“lignite”, “brown coal”, `generation__fossil_brown_coal_lignite`	`fuel=lignite`	Align to ENTSO-E naming
“blackout”, “grid outage”, “supply interruption”	`TemporalEvent=BlackoutEvent`	Event class
“policy change”, “regulatory decision”, “ministerial decree”	`TemporalEvent=PolicyChange`	Event class
“weather heatwave”, “high temperatures”, “καύσωνας”	`Theme=Weather`	Theme + optional sub-type
“Greece bidding zone”, “IPTO”, `10YGR-HTSO-----Y`	`Zone(name='Greece')`	Zone/MapCode mapping
“PPC”, “Public Power Corporation”, “ΔΕΗ”	`Entity(name='PPC')`	Organization synonyms
“Regulator”, “RAE”, “Hellenic Energy Regulator”	`Entity(type='Regulator')`	Organization class

Table 2-1: vocabulary normalization examples

In tandem with ontology context, we impose schema-constrained generation to ensure the output format is strictly structured. We use JSON Schema as a contract for the LLM's output. The JSON Schema defines the expected structure (keys and data types) for each type of object in the ontology. For example, for an Event extraction we might require keys: "type", "description", "date" (ISO 8601 format), "location" (canonical region name or code), "entities_involved" (array of entity identifiers or names), and "source" (the article URL or ID). This schema (provided in the prompt or as an instruction) acts as a specification that the model should follow. The model is instructed to output only valid JSON conforming to this schema and nothing else. This approach leverages the fact that modern LLMs can be guided to follow a format meticulously if the prompt is clear. By including the schema (or a snippet of it) in the prompt, we effectively whitelist JSON keys and value types. The model learns that any output must fit that template, reducing the chance of free-form digressions or missing fields.

An additional measure is the use of grammar-constrained decoding: decoding the LLM's output with a formal grammar that only generates syntactically correct JSON. Research has shown that applying a context-free grammar during generation can prevent invalid tokens and guarantee well-formed JSON output[4],[5]. In practice, this might involve using a tool or library that wraps the LLM and stops it from producing characters that violate the JSON structure (e.g. an unmatched brace). Grammar-constrained generation complements the prompt-level schema hints by enforcing correctness at decoding time (it ensures syntax validity, while the JSON Schema ensures semantic validity of keys/values). The benefit is that we rarely get malformed outputs – a common failure mode when asking LLMs to output complex JSON is accidentally producing extra commas or commentary text. With a grammar or incremental JSON validator, those are eliminated. The downside to grammar-constrained decoding is the added complexity: not all LLM APIs natively support grammars, so it can require custom decoding logic. Also, a grammar can guarantee form but not that the content is correct (the model could still place a value in the wrong field) [4],[5]. Therefore, we still rely on full JSON Schema validation after generation as a safety net.

In summary, ontology-aware prompting and schema constraints greatly enhance reliability. The theoretical foundation comes from principles of controlled text generation and structured prediction. By giving the model a schema to adhere to, we turn the generation into a kind of structured fill-in-the-blanks task rather than open-ended text writing. The model's creativity is

thus channeled strictly into filling factual values from the input text into the correct JSON slots. If the model attempts to produce something outside the schema, the validator or grammar will catch it, and we can discard or correct that output. This approach draws inspiration from ideas in semantic web and databases: our ontology and schema play a role analogous to an OWL/SHACL schema in RDF systems, defining what constitutes a valid fact, and the LLM is constrained to generate only those valid facts. The result is a schema-conforming JSON output ready for ingestion into the knowledge graph.

2.3.5. Temporal Anchoring and Geospatial Grounding

Energy events are inherently tied to when and where they occur. Thus, a robust knowledge extraction process must normalize temporal and geospatial references from text. Temporal anchoring is the practice of converting date/time expressions in text into a standard, machine-readable format (ISO 8601 timestamps). For example, if an article says "late July 2025", the system should interpret this, perhaps as an interval like "2025-07-20/2025-07-31" (if the exact date isn't given). We attach such normalized timestamps to the extracted events or facts. Using ISO 8601 (e.g. "2025-07-24T00:00:00Z" for July 24, 2025) ensures that all events on the timeline can be compared and ordered chronologically. In cases of ambiguous or relative time expressions – e.g. "yesterday" or "last week" – the extraction process needs to use the article's publication date as a reference. For instance, "yesterday" in a news piece published on 2025-08-01 would be anchored as 2025-07-31. If an exact date cannot be determined, we allow an uncertainty range or mark the timestamp with a qualifier (e.g. an approximate_date flag or using the start of the mentioned month when only month/year are known). The theoretical basis here comes from temporal information extraction research (e.g. TIMEX3 in NLP) – the idea is to attach every event or fact to a timeline consistently. By doing so, our downstream system can align events with time series anomalies by matching timestamps. In the prompt design, we instruct the LLM to output dates in ISO format when possible (and our post-processing will handle parsing of any free-text dates). We also incorporate a Temporal Ontology (as part of our schema) that defines concepts like TimeInstant or TimeInterval to standardize how time is represented in the extracted JSON.

Geospatial grounding is analogous, focusing on the "where." Locations mentioned in text can be ambiguous or vary in granularity. Our approach is to map textual location mentions to canonical geographic identifiers. For example, an article might mention "Μακεδονία" which could refer to a region in Greece (Western Macedonia, Central Macedonia, etc.) or generally the historical region. We resolve this by using a predefined list of canonical regions and country names (for instance, the ISO country codes or a standardized list of admin regions in Greece). If the text says "Attica" or "Αττική", we map it to a canonical identifier for the Attica region. If a specific city or facility is mentioned (e.g. "Λαύριο Power Station"), the system should ground it to known entities in the ontology (like a node representing that power station with coordinates). This grounding often requires an external gazetteer or lookup table – essentially an algorithmic step to disambiguate place names. In theoretical terms, it's related to entity linking (linking a text mention to a knowledge base entry). We include in our pipeline a geospatial canonicalizer that consults a database of known energy locations (possibly using a PostGIS-backed repository of power assets and regions) to find the best match. For the LLM extraction stage, we also provide guidance: the prompt might say "identify any locations and use canonical region names". If the LLM is uncertain (e.g. the text says "in the north of the country" without naming it), the model should output a generic "location": "Greece (unspecified region)" or a null value for location with a note. We prefer not to guess – abstaining or using an "unspecified" placeholder is better than mislinking to the wrong place.

The combination of temporal anchoring and geospatial grounding gives each extracted event a clear when and where context in a standard form. This is crucial for aligning with time series anomalies: an anomaly has a timestamp and possibly a region (e.g. a spike in Greek national demand vs. Attica regional demand). We can only match an event to an anomaly if both share the same normalized time and location reference. By enforcing these normalizations at extraction time, we avoid ad-hoc string matching later. Conceptually, this part of the system echoes known standards like ISO 8601 for time and GeoSPARQL/ISO country codes for location in semantic data integration. It ensures that the knowledge graph we build has a consistent temporal and spatial index – enabling queries like "find events within 1 day and in the same region as anomaly X." All told, temporal and geospatial normalization are indispensable for grounding the extracted knowledge in real-world coordinates of time and space.

2.3.6. Prompt Design Principles for Reliable Structured Output

Designing the prompt for the LLM is both an art and a science. Our prompts are carefully structured to maximize extraction accuracy while minimizing irrelevant text. Generally, the prompt includes multiple components, each serving a specific purpose:

- **System and Task Instruction:** A concise description of the task, e.g. "You are an information extraction system that reads news and outputs facts in JSON. Only produce valid JSON according to the schema. Do not include explanations." This sets the overall behavior: the model knows it must produce a particular format and nothing extra.

- **Ontology and Schema Excerpt:** We provide a summary of the relevant ontology classes and relationships expected. For instance: "Relevant event types: Blackout, Maintenance, PolicyChange, WeatherEvent. Relevant entity types: PowerPlant, Company, GovernmentAgency, Region. The output JSON schema has fields: type, date, location, entities_involved, description, source." By giving this, we narrow the context: the model will focus only on extracting those types of information and use those exact field names. This acts as a whitelist of what it should talk about.

- **Few-Shot Examples (especially in Greek):** We include one or more example extractions, since our articles are primarily in Greek. For example, we might show a short Greek sentence and a correct JSON output. E.g.: «Στις 12 Αυγούστου, ένας κεραυνός προκάλεσε μπλακ άουτ στη Θεσσαλία.» with a following JSON: {"type": "BlackoutEvent", "date": "2025-08-12", "location": "Thessaly, GR", "cause": "Lightning strike", "entities_involved": [], "source": null}. This demonstration helps the model understand exactly how to format the output for real inputs. It also signals that Greek names and text should be handled (the model sees Greek in the prompt, reducing confusion when the input is Greek).

- **Output Constraints and Guardrails:** We explicitly remind the model to output JSON only. Phrases like "If a field is unknown or not mentioned, use null or an empty array. Do not add any commentary. Do not deviate from the schema." are included. We also emphasize provenance: "Include the source field with the article URL provided" (if

the URL or ID of the article is known to the model or passed in). These guardrails ensure that the model doesn't, for example, start explaining its reasoning or outputting text outside the JSON. The instruction to use null for unknowns is important to prevent the model from guessing or hallucinating values – it is effectively an abstain mechanism.

All these components are combined in a logical order: typically system role instruction, then schema/ontology description, then examples, then the actual article text (as the prompt input to be processed). The prompt is thus quite structured, and this structure is designed following best practices from prompt engineering research and our own trials. One principle is to place the example demonstrations right before the task input, to leverage in-context learning: the model is more likely to mirror the format it just saw in the examples. Another principle is clarity and brevity – we don't include any superfluous text that might confuse the model. Every part of the prompt either defines the format or exemplifies the task.

For reliability, we also instruct the model on what to do when uncertain. For instance, "If the article does not mention a specific date, set date to null. Do not fabricate a date." These instructions act as fabrication guardrails, critical in preventing the model from introducing false information. We prefer null or "unknown" markers over wrong guesses, maintaining high precision at slight cost of completeness. In an academic sense, this prompt design embodies the concept of schema-guided generation: the model is virtually guided by a mini specification (the ontology) and must conform to it. Past work has shown that such constrained prompts greatly improve the quality of structured outputs from LLMs. Our approach aligns with those findings, combining them with domain-specific adjustments (bilingual examples, etc.).

2.3.7. Reliability, Validation, and Evaluation of Extraction

Even with careful prompting, the system must handle errors gracefully and measure its own performance. We incorporate multiple layers of reliability checks:

JSON Schema Validation: After the LLM outputs a JSON, we run a validator against the official schema (which defines allowed classes, fields, data types, etc.). Any output that fails validation is rejected or flagged. For example, if the model returns an unknown field "location_name" instead of the expected "location", or if a required field is missing, the

validation will catch it. This is crucial to avoid ingesting malformed data into the knowledge graph. Invalid outputs are placed into an "unmapped queue" for manual review or re-processing. This ensures ingestion safety – only clean, structured data enters our database.

Deterministic Generation: To enhance reproducibility, we typically run the LLM with deterministic settings (e.g. temperature 0) when extracting facts. This reduces variability – the same input should yield the same output each time. It also prevents the model from randomly drifting in style or structure between runs. If nondeterministic methods are used (for diversity), we ensure the critical fields remain consistent or we take multiple outputs and choose the best via validation.

Error Handling and Abstention: We instruct the model to abstain when unsure (outputting nulls or empty lists as mentioned). If the model truly doesn't follow the format or produces a non-JSON answer, our system treats it as a failure case. We might then fall back to a simpler prompt or a smaller extraction scope. Categorizing errors helps in debugging: we maintain an error taxonomy including (a) Malformed JSON (caught by parser), (b) Schema violation (JSON structure okay but wrong content types), (c) Semantic errors (e.g. the model filled a field with plausible text that's not actually in the article). The last type is the hardest to catch automatically – it requires spot-checking. But we mitigate it by emphasizing provenance: every extracted fact is supposed to be verifiable in the source text. The JSON could even carry a snippet or reference to the exact sentence it came from.

Evaluation and Quality Metrics: We evaluate the extraction quality both intrinsically and extrinsically. Intrinsically, one signal is schema coverage – does the model populate all fields it should when information is present? Also, do the outputs cover the range of classes and relations we expect from the corpus? For instance, if our 250 articles include many on weather, we expect many WeatherEvent outputs; if not, perhaps the model is missing them. We monitor the distribution of extracted classes and compare it to known frequencies (if available) or expectations. We also evaluate precision by spot-checking a sample of extracted facts against the source text: are the named entities correct? Did the model invent any detail? Spot-checking focuses on critical classes (e.g. facts that would be used in explanations of anomalies). If an important category like "BlackoutEvent" has low precision (many false extractions), we adjust the prompt or refine the ontology definitions to the model.

When possible, we assemble a small "gold set" of articles with manually annotated facts. We then compare the LLM's output to this gold data to estimate precision and recall. This gives quantitative scores (like an F1 measure) for the extraction. However, building a large gold set can be labor-intensive, so our emphasis is on high precision through design and then ensuring reasonable recall by broad prompt coverage.

Drift Monitoring: Over time or with different model versions, the output might drift (e.g. a new model update might format things differently or use synonyms). We keep an eye on this by periodically re-validating a fixed set of example inputs. If a drift is detected (say the model starts using a non-canonical location name or forgets to include citations), we intervene with prompt adjustments or additional fine-tuning of instructions. Since our pipeline might be running on updates from news over time, maintaining consistency is important – otherwise the knowledge graph could get inconsistent entries.

In conclusion, the theory behind our LLM-based knowledge extraction is to combine the strengths of generative models (flexibility and understanding) with the rigor of schema enforcement and validation. By doing so, we obtain a system that can ingest diverse unstructured inputs and output rich structured knowledge, all while minimizing the risk of error. The careful prompt design, use of ontologies, and validation steps ensure that the extracted facts about the energy domain are accurate, normalized in time and space, and ready to support higher-level reasoning like anomaly explanations. This lays a solid theoretical foundation for the implementation in Chapter 4 and the explainability techniques discussed next.

2.4. Knowledge Graphs and GraphRAG for Explainability

2.4.1. Graph-Structured Knowledge for Anomaly Explanation

Knowledge graphs are an effective way to represent heterogeneous information – they store entities (nodes) and relationships (edges) in a flexible, connected structure. In the context of energy systems, a knowledge graph can link together many data types: time series signals (loads, generation), detected anomalies in those signals, events extracted from news, key entities like power plants or companies, thematic factors like weather or policy changes, and so on. The motivation to use a graph structure for anomaly explainability stems from the need for

multi-hop reasoning and provenance-rich context[7],[8]. An anomaly (for example, a sudden drop in electricity demand on a certain date) might be explained by an event (e.g. a large outage or a public holiday) that is not directly obvious from the data alone. By storing events and facts in a graph, we can traverse connections: from an anomaly node we can find related events that occurred around the same time and place, then from those events we can find involved entities or broader themes (like "heatwave" or "grid maintenance"). This multi-hop traversal is something graphs excel at – you can find paths and neighborhoods of information that connect seemingly disparate facts.

Crucially, every piece of information in the graph can carry its provenance (where it came from), enabling trustworthy explanations. Rather than a black-box model guessing the cause of an anomaly, we have a graph that explicitly links an anomaly to evidence (news, events) that can be cited. This structured approach addresses one of the core challenges in explainable AI: the need to justify why something happened with verifiable sources. The graph acts as a knowledge base that the explanation system can draw from, ensuring that answers about anomalies are grounded in data or documented events, not just learned patterns.

In theory, using a graph for anomaly explanation aligns with the retrieval-augmented generation paradigm, where an external knowledge source is used to inform an LLM's output. Instead of retrieving documents by keywords, we retrieve a tailored subgraph of relevant facts. Graphs are particularly well-suited for global context and sensemaking. They provide a holistic view: for example, an anomaly on date D in region R can be connected to all events on D in R (or even events in neighboring regions, if the graph encodes power grid connectivity). This comprehensive neighborhood can reveal patterns (maybe D was a national holiday affecting all regions). If we relied only on individual text snippets, we might miss the big picture or fail to connect two related events reported in different articles. A graph naturally merges information from multiple sources because they share nodes (e.g. the same region node linked to multiple events). In summary, the knowledge graph offers a structured memory of the system's factual context, which is invaluable for explaining anomalies through multi-hop, multi-source reasoning.

2.4.2. Property Graph Model and Ontology Alignment

Our system uses a property-graph model, conceptually similar to Neo4j's data model. In a property graph, nodes have labels and properties (key-value attributes), and edges have types and can also carry properties. This model is flexible and expressive, allowing us to model the energy domain ontology naturally. We define several key node types (labels) in the graph, each corresponding to a concept from our ontology:

- **TimeSeries:** representing a stream of measurements (e.g. the national power demand, the solar generation of a specific plant). A TimeSeries node might have properties like metric_type (load, generation, price, etc.), zone (the region or country it pertains to), and maybe an identifier linking to the ENTSO-E source.

- **Anomaly:** representing a detected anomaly on a time series. An Anomaly node could have properties such as timestamp (when it occurred), severity, expected_value vs actual_value, etc. It is typically linked to the TimeSeries it belongs to.

- **Article:** representing a news article or report document. It has properties like published_date, source_url, and perhaps a short title or reference.

- **TemporalEvent:** representing an event that occurred, extracted from text. This corresponds to events in our ontology (like BlackoutEvent, WeatherEvent, PolicyChange, etc.). Properties include type (the subclass of event), description (a brief description from the text), date (normalized date or interval), and source_ref (linking back to an Article or source).

- **Entity:** representing real-world entities such as organizations, facilities, persons, or other actors. For example, a node for Public Power Corporation (PPC) or a node for a specific power plant. These nodes often come from the ontology's actor or asset classes and have properties like name, entity_type and possibly external IDs.

- **Theme:** representing abstract themes or categories relevant to events, such as Weather, Market, Regulatory, Technical, etc. A theme node is like a tag that can be attached to events or anomalies indicating the nature of their cause (e.g. an event might be tagged with theme "Weather" if it's a heatwave).

- **Zone/Region:** representing geospatial areas (countries, regions, etc.). For instance, a node for Greece, and nodes for its sub-regions (Attica, Macedonia, etc.). These often come from a location ontology or simply a predefined set of region nodes with properties like name and standardized codes.

Correspondingly, we define relationship types to connect these nodes in meaningful ways some of which are displayed below:

- **HAS_ANOMALY:** a relationship from a TimeSeries node to an Anomaly node. This indicates that the anomaly occurred in that particular time series. The edge might have a property like window or date to indicate when.

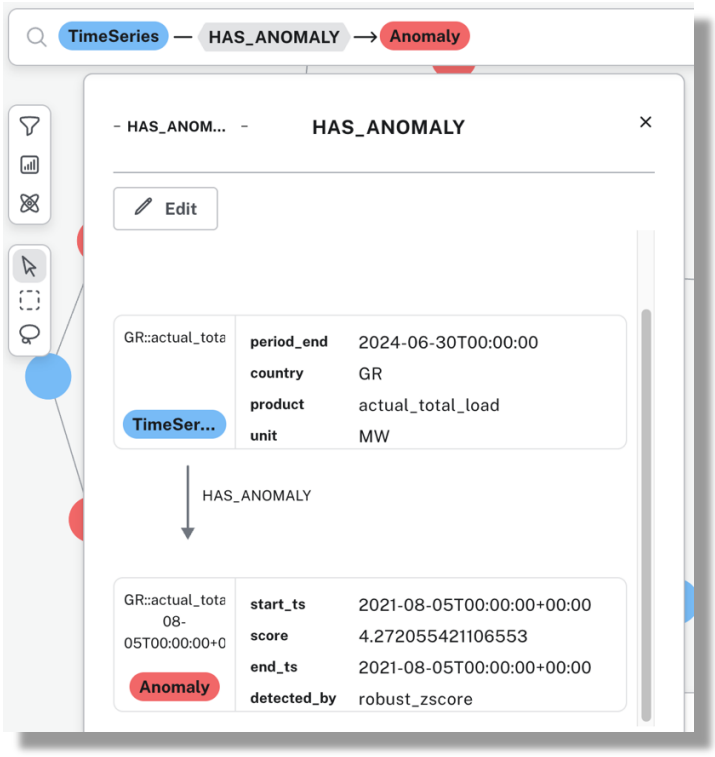


Figure 2-3: HAS_ANOMALY relationship screenshot

- **MENTIONS:** linking an Article to an Entity or to a Theme. If a news article mentions the "Public Power Corporation", we create an edge Article —MENTIONS→ Entity(PPC). This way we know which entities were involved or quoted in that article. Likewise, if the content of the article is about a heatwave, we could link the Article —MENTIONS→ Theme(Weather).

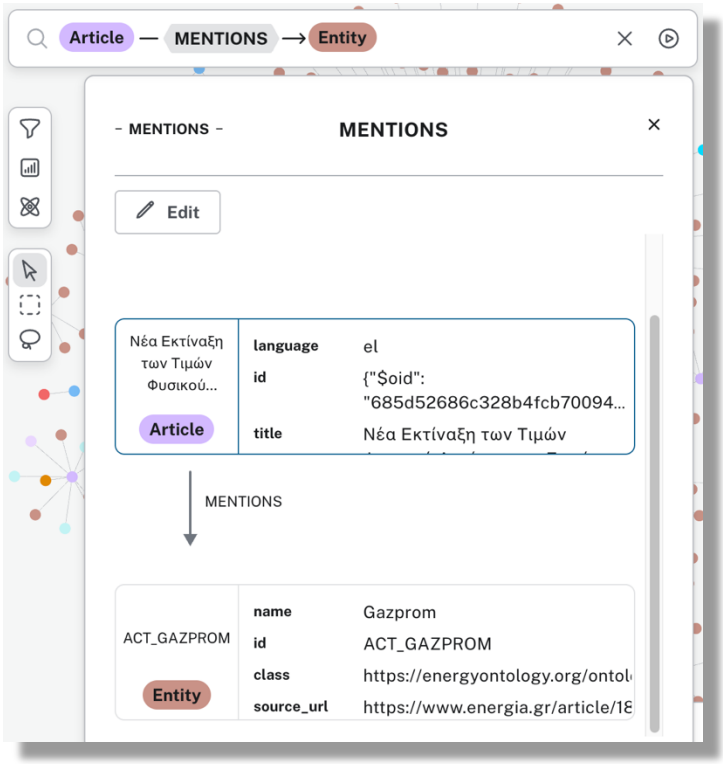


Figure 2-4: MENTIONS relationship screenshot

- **TAGGED_THEME:** linking a TemporalEvent to a Theme. For instance, an event node for "Heatwave in July 2025" could be TAGGED_THEME→ Weather. Or a policy change event might be tagged as Regulatory or Market theme. This provides a way to categorize events by general type.

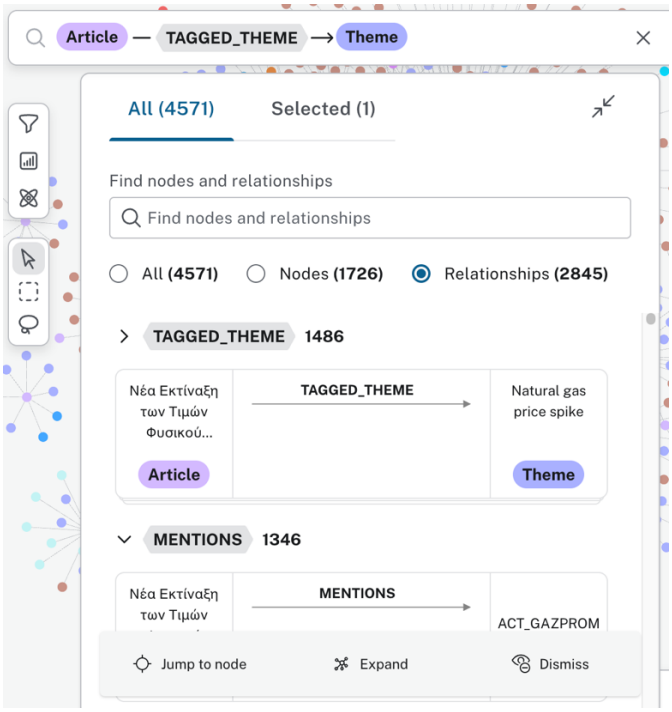


Figure 2-5: TAGGED_THEME relationship screenshot

- **LOCATED_IN:** capturing geography, this relationship connects events or entities to a Zone. An event that happened in Attica would have TemporalEvent —LOCATED_IN→ Zone(Attica). Similarly, a power plant entity could be LOCATED_IN a certain region. This relation anchors nodes to the spatial hierarchy.

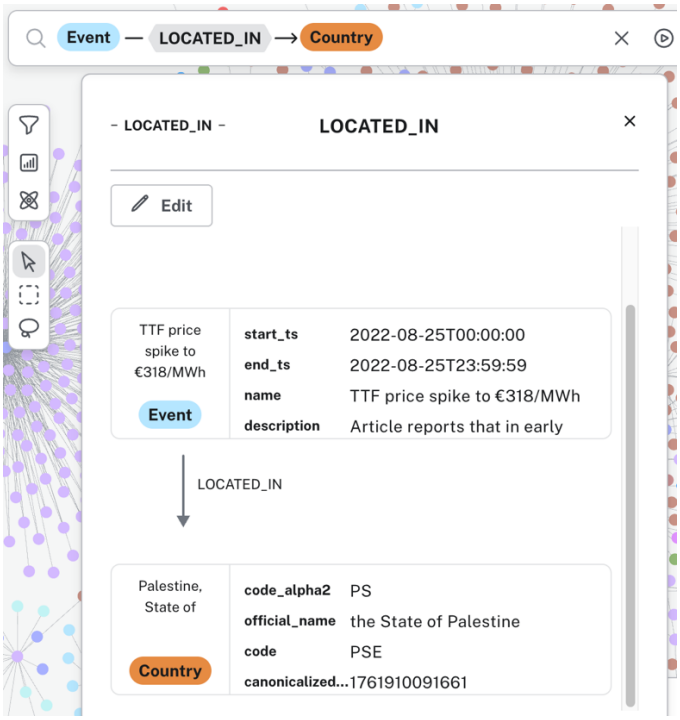


Figure 2-6: LOCATED_IN relationship screenshot

Additionally, we structure time using a time index in the graph. A common technique is a time-tree: we create nodes for Year, Month, Day, etc., and link events or anomalies to those. For example, an Anomaly on 2025-07-24 might have an edge to a Day node representing "2025-07-24", which links up to a Month node "2025-07" and a Year node "2025". This time-tree allows efficient range queries (find all events in July 2025 by traversing that node). It's an implementation detail, but conceptually it underscores that time is a first-class dimension in our knowledge graph.

The entire graph is aligned with the ontologies we defined (recall the Temporal, Energy, Event ontologies from Section 2.3). Each node type corresponds to an ontology class, and each relation type corresponds to a defined relationship in our schema. By using a property graph (Neo4j) to store this, we get the advantage of indexing and query language (Cypher) to easily retrieve subgraphs of interest. The property graph model is appropriate here because of its natural support for property-rich nodes (e.g. we can store the full text of an article as a property

of the Article node if needed, or store numeric values in Anomaly nodes). It's also intuitive for developers and analysts. This design is distinct from an RDF triple store; however, it's conceptually similar in that the ontology provides a schema for nodes and edges. In practice, Neo4j was chosen as the conceptual backbone due to its speed in graph traversal and its ecosystem for building graph-powered applications. We ensure that all data ingested (from LLM extraction or time series) respects the ontology definitions, thus the graph remains semantically consistent. This consistent property graph will be the substrate on which we perform explainability through GraphRAG.

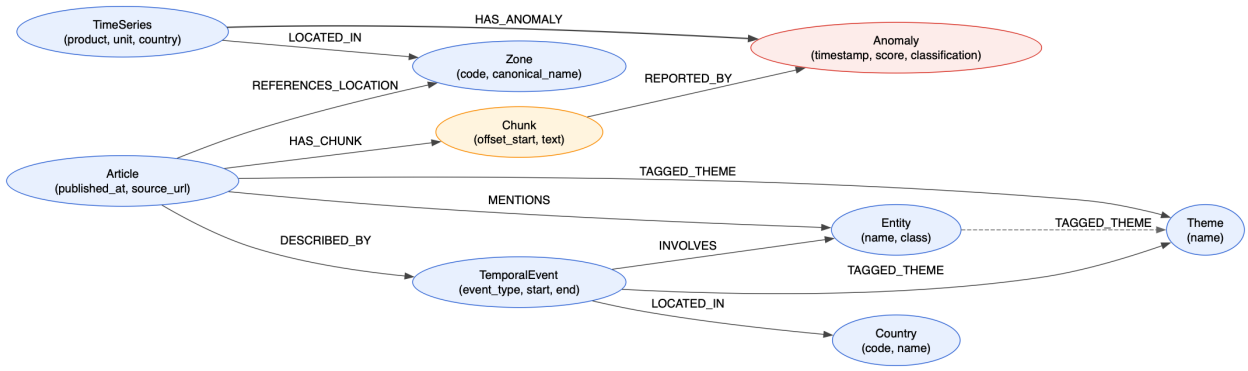


Figure 2-7: Graph Schema for Neo4j

2.4.3. Provenance and Trust in the Knowledge Graph

An explainable system must establish trust with the analyst user. We achieve this by weaving provenance information throughout the knowledge graph. Every fact in the graph is traceable back to its source:

- **Source URLs and Documents:** Article nodes carry the actual source URL or reference (e.g. an ID linking to a full-text repository). When an event node is extracted from an article, that event node has a property or link indicating the source article. For instance, TemporalEvent "Heatwave on July 24" might have a property source_url: "https://example.com/news123" and/or an edge to the Article node from which it was extracted. This means if the analyst questions the explanation, they can follow the link to read the original article in full.

- **Timestamps and Ingestion Info:** Each node/edge added from extraction can have metadata like extracted_at (the time our system added it) and model_version or prompt_version (which LLM or prompt template was used). This provenance metadata is useful for audit – if an error is later found in an extraction, we know which model/prompt likely caused it.

- **Citation Strings:** We sometimes store a short citation or reference snippet with events. For example, an event node might include a property citation_text like "(Reuters, 24/07/2025)" to be used in generated explanations. This is not strictly necessary in the graph, but it's convenient to have a pre-formatted reference for use in the output.

- **Confidence scores:** Though our extraction is largely zero-shot, if we have any notion of confidence (say from a validation heuristic), it could be stored. For instance, if multiple articles report the same event, we might increase confidence. Or if an event was extracted but with missing fields, we mark it as low confidence and that could be considered during explanation generation (maybe requiring a caveat or cross-check).

By attaching provenance at the node and edge level, the graph becomes a transparent knowledge base. This addresses a common concern: LLMs can hallucinate, but if our explanation is built from graph facts that are each source-backed, the final output can include explicit citations. In effect, the knowledge graph and its provenance data allow the LLM to become a truthful narrator rather than a creator of new claims. Each relationship used in an explanation (e.g. linking an anomaly to a cause event) is something stored in the graph because it was observed or derived from data. We also model provenance in relationships: for example, if we assert Anomaly —DESCRIBED_BY→ Event, that edge might carry a supporting_source attribute (maybe pointing to an article that mentioned both the anomaly and event, if such exists).

This approach is conceptually aligned with the W3C PROV-O (Provenance Ontology) principles, where every entity and relationship can have associated provenance. While we don't explicitly implement a full PROV ontology in Neo4j, we adhere to the idea: who/what/when was this piece of knowledge generated. For trust, this is crucial; an analyst can drill down from a conclusion ("the anomaly was caused by X") to the evidence nodes ("X was a heatwave event recorded in this article, here's the link"). The presence of explicit provenance also aids the LLM

when generating explanations: we actually pass these sources into the prompt so the model can cite them in the text. This practice of including sources is known to improve the factual accuracy of generated answers and provides users the ability to verify each claim.

In summary, the knowledge graph is not just a collection of data points; it's a web of evidence. Each node and edge is backed by some origin, and we store that link. This builds user confidence that explanations are traceable. It also helps maintain the integrity of the system – errors can be traced and corrected at the source level. By designing the graph with provenance in mind, we ensure that explainability is a first-class property, not an afterthought.

2.4.4. Cypher-Driven Retrieval of Contextual Evidence

To explain an anomaly, we need to gather the right context from the graph. This is fundamentally a retrieval problem: given an anomaly node (or an anomaly description), find the most relevant connected information in the graph that could explain it. We take a Cypher-first retrieval approach, meaning we leverage structured graph queries (in Neo4j's Cypher query language) to fetch a subgraph of evidence before invoking any language model to generate an explanation. The retrieval is guided by a few key parameters based on the anomaly's attributes:

- **Temporal Window:** We constrain the query to events that occurred near the anomaly's time. For instance, if the anomaly is at 2025-07-24 18:00 (an evening spike), we might query for events on the same day or within ±1 day of that timestamp. The window can be adjusted depending on anomaly duration; e.g., for a multi-day anomaly, we look across that span. This ensures we only retrieve events that could reasonably be contemporaneous causes or context. (A heatwave a year earlier likely isn't relevant, so we exclude it.)

- **Geographic Scope:** We filter or prioritize events and facts by location. If the anomaly is specific to the Attica region's grid, we first look for events located in Attica. If not much is found, we might broaden to national-level events (Greece) or neighboring regions, but with lower priority. Similarly, if the anomaly is on a cross-border interconnection, we consider events in either of the countries involved. The graph's

LOCATED_IN edges make this filtering straightforward: a Cypher query can match patterns like (e:TemporalEvent)-[:LOCATED_IN]->(:Zone {name: "Attica"}).

- **Ontology Class Filters:** Depending on the type of anomaly or the domain knowledge, certain event types or themes might be more relevant. For instance, if the anomaly is a demand drop, we expect relevant events might be of type WeatherEvent (like a cooling event or storm) or InfrastructureFailure. If it's a price spike anomaly, maybe PolicyChange or MarketEvent nodes are more pertinent. We can encode these expectations in the query or simply retrieve all events and later rank them by a heuristic. In practice, we might label some events as high-priority (e.g. any Blackout event is very likely important for grid anomalies). The ontology allows us to do this systematically.

- **Graph Connectivity:** We often exploit the graph structure by performing expansive queries. For example, find any events within time window that are in the same region or that involve an entity related to the anomaly. If the anomaly node itself is linked to something (say an Anomaly is linked to a particular PowerPlant entity, if the anomaly was detected specifically at that plant's output), we include events involving that same entity. Cypher queries can traverse multiple hops, e.g., "find events that involve any entity that is connected to this anomaly." This way if the anomaly is at Plant X, and we have an event "Plant X went offline due to maintenance", the query will catch that event via the shared entity.

After defining the query criteria, we execute Cypher to retrieve a set of candidate nodes and relationships – essentially an evidence subgraph. For example, the query result might include two event nodes (a heatwave and a local outage), one theme node (say "Weather"), and their links to region or entity nodes, plus the article nodes that describe those events. This subgraph is the raw material for the explanation.

Because multiple pieces of evidence may be found, we consider lightweight ranking heuristics to decide which ones to highlight or include. Some heuristics: - Temporal proximity: An event that happened just hours before the anomaly likely ranks higher than one that happened five days earlier. - Frequency/Multiplicity: If a type of event is common, it might be less explanatory than something rare. But if multiple sources all point to the same explanation,

that boosts confidence (e.g. three different news articles all mention a heatwave). - Causal relevance (by class): A priori, we might weight weather events as more explanatory for demand anomalies, and grid outages more for supply anomalies, etc. This introduces domain knowledge into ranking. - Connectivity: If an event is directly linked to the anomaly node (for instance, in the graph we might directly link a known cause to the anomaly), that should obviously be included. Our system currently doesn't create a direct cause edge automatically, but if it did or if an analyst tagged it, that would get top priority.

The retrieval step is done in the graph database because it can use indexes and relationships efficiently. This is more precise than doing a vector search over text: we leverage the structured relationships to get relevant context with high precision. It's also explainable in itself – we can log the query and see why certain nodes were returned (due to matching time and location, for example).

By performing a Cypher-first retrieval, we reduce the load on the LLM. Instead of handing the LLM a massive trove of documents or expecting it to recall facts from parameters, we give it a concise set of extracted facts from the graph. This improves both speed and reliability. It's worth noting that this approach is deterministic given the same graph state and query parameters – the same anomaly will always retrieve the same evidence set, which is good for consistency. One could tweak the Cypher queries and see directly how the result set changes, which is more transparent than adjusting a fuzzy similarity threshold in vector retrieval.

In theoretical terms, our retrieval strategy aligns with how GraphRAG is conceptualized: first retrieve a connected subgraph relevant to the query (here the "query" is essentially: *"Why did anomaly X happen?"*), then feed that to the generative model. It's a knowledge-driven form of retrieval augmented generation.

2.4.5. Graph-Augmented Generation of Explanations (GraphRAG)

GraphRAG refers to Graph-based Retrieval-Augmented Generation. In our scenario, after retrieving the subgraph of evidence via Cypher, we use it to augment the prompt of the LLM that will generate the explanation. The idea is to provide the LLM with structured context so that it can produce a grounded answer that includes the relevant facts and cites sources. This

approach drastically reduces hallucinations, because the model doesn't have to invent any explanation – it sees actual events and data points from the graph to base its reasoning on.

The assembly of the prompt for explanation follows a structured format similar to how we did for extraction, but with a focus on explanation. We include sections like: **- Anomaly Overview:** We start by describing the anomaly in natural language, including its key details. For example: "An anomaly was detected on the electricity demand time series for Greece on 24 July 2025. The demand spiked 15% above forecast during the evening peak." This gives the LLM and eventually the reader a clear statement of what needs explaining. **- Retrieved Evidence (Events and Details):** We then present the relevant facts from the graph. We might format this as bullet points or a short paragraph per item. For instance: "Evidence: (1) A severe heatwave was reported in Greece on 23–26 July 2025, with record high temperatures (source: [NewsArticle1]). (2) The government declared a public holiday on 24 July 2025 in some regions due to emergency measures (source: [NewsArticle2]). (3) Increased use of air conditioning was widely reported (Theme: Weather)." Each piece of evidence corresponds to nodes in the subgraph (events or themes), and we attach a citation in brackets referencing the source article node. In practice, we ensure each evidence sentence contains the essential info and a [Citation]. **- Entity and Theme context:** If relevant, we add one-liners about key entities or themes. E.g. "Public Power Corp (PPC) reported no infrastructure failures on that day [CompanyReport]." Or "This anomaly occurred during a known period of high temperatures (Weather event)." This is optional and depends on what the graph has – sometimes the evidence itself suffices. **- Instruction and Answer Formatting Tips:** After listing the evidence, we often include a brief instruction to the model: "Using the above information, explain the likely reasons for the anomaly. Your answer should be a concise paragraph. Cite the sources for each factual claim. If the evidence is insufficient or contradictory, note that the cause is uncertain." This guides the LLM on how to compose the final answer. We stress the use of citations and mention the possibility of uncertainty so the model knows that "unknown" is an acceptable answer if warranted.

With this prompt, the LLM then generates the explanation. Since it has explicit facts and references in front of it, it tends to stay faithful to them – effectively performing a summarization or reasoning task over a mini-knowledge-graph. Each piece of evidence in the

prompt is like a node or triple that the model can incorporate into a narrative. For example, the model might produce: "The anomaly on 24 July 2025 can be explained by an extreme heatwave that drove up electricity demand. During 23–26 July 2025, Greece experienced record high temperatures, leading to surging air conditioning use[2]. The spike occurred on a day that was even declared an emergency public holiday, which indicates widespread impact. Thus, the demand surge was not a random fluctuation but correlated with a weather-induced demand increase." (And the citations and here correspond to the evidence sources given, demonstrating provenance.)

Faithfulness is a key evaluation criterion for the LLM's answer. Because we structure the input, we expect the output to contain only claims derivable from the evidence. If the model tried to add something not supported (hallucination), ideally it would not find it in the prompt and thus be less likely to mention it. We also explicitly told it to say "unknown" if it can't find a reason. For example, if no events were found or nothing obvious, the model might answer: "The cause of the anomaly on that date is unclear; no significant events were reported around that time, so it may have been due to unknown factors." This kind of answer is important – it's better to acknowledge uncertainty than to guess.

Our GraphRAG approach can be contrasted with a pure text RAG: normally, one might vector-search for relevant passages in news articles and feed them to the LLM. We instead retrieve by graph (structured) and feed in a distilled form of knowledge. In some cases, we may complement this with actual text snippets (for richness, especially if a quote or detail is needed). A hybrid approach could embed both the structured data and a short quote from an article in the prompt. For instance, an event node might carry a short description "Heatwave with 40°C temperatures" which we include verbatim. If needed, we could also retrieve the top relevant paragraph from an article via embedding search and show it. This hybrid Graph+Text RAG can improve completeness when the graph node alone is too abstract. But importantly, even that text snippet is anchored to a node, maintaining the structured context.

By using GraphRAG, we also reduce hallucinations and enforce relational accuracy. The graph structure ensures that the relationships (e.g. cause-effect, location-event) are explicitly given, so the model is less likely to make an incorrect connection. For example, if two events

are unrelated in the graph, we won't present them as linked evidence, so the model won't incorrectly tie them together.

This approach draws on recent research: Edge et al. (2024) demonstrated that graph-based RAG yielded more comprehensive and diverse answers to broad analytical queries than standard RAG[2]. Our use case is a specific form of query (anomaly explanation), but we similarly find that pulling in a connected subgraph (anomaly, related events, entities) gives the model a richer context to answer the "why" question. Essentially, GraphRAG lets the LLM do what it's good at (language and reasoning) while the knowledge graph does what it's good at (storing and retrieving structured relations). The combination ensures that the explanations are both grounded in data and articulated in a coherent narrative.

CHAPTER 3

3. SYSTEM ARCHITECTURE

3.1. System Overview

The system follows a layered architecture that maps cleanly to microservices. Guided by the requirements of Chapter 2, we separate responsibilities into five layers: Data Acquisition, Storage & Preprocessing, Semantic Extraction, Knowledge Graph, and Application. This separation keeps concerns isolated and allows each layer to evolve independently (e.g., add a data source, change an extraction method, or scale a service) without ripple effects.

- Data Acquisition pulls structured metrics (ENTSO-E) and unstructured text (news/RSS).
- Storage & Preprocessing stages inputs into weekly aggregates, exports CSVs, and prepares payloads.
- Semantic Extraction turns raw artifacts into entities, events, themes, and anomalies.
- Knowledge Graph (Neo4j) integrates all semantics under consistent time and location references.
- Application exposes APIs and UIs (timeline views, GraphRAG answers) to end users.

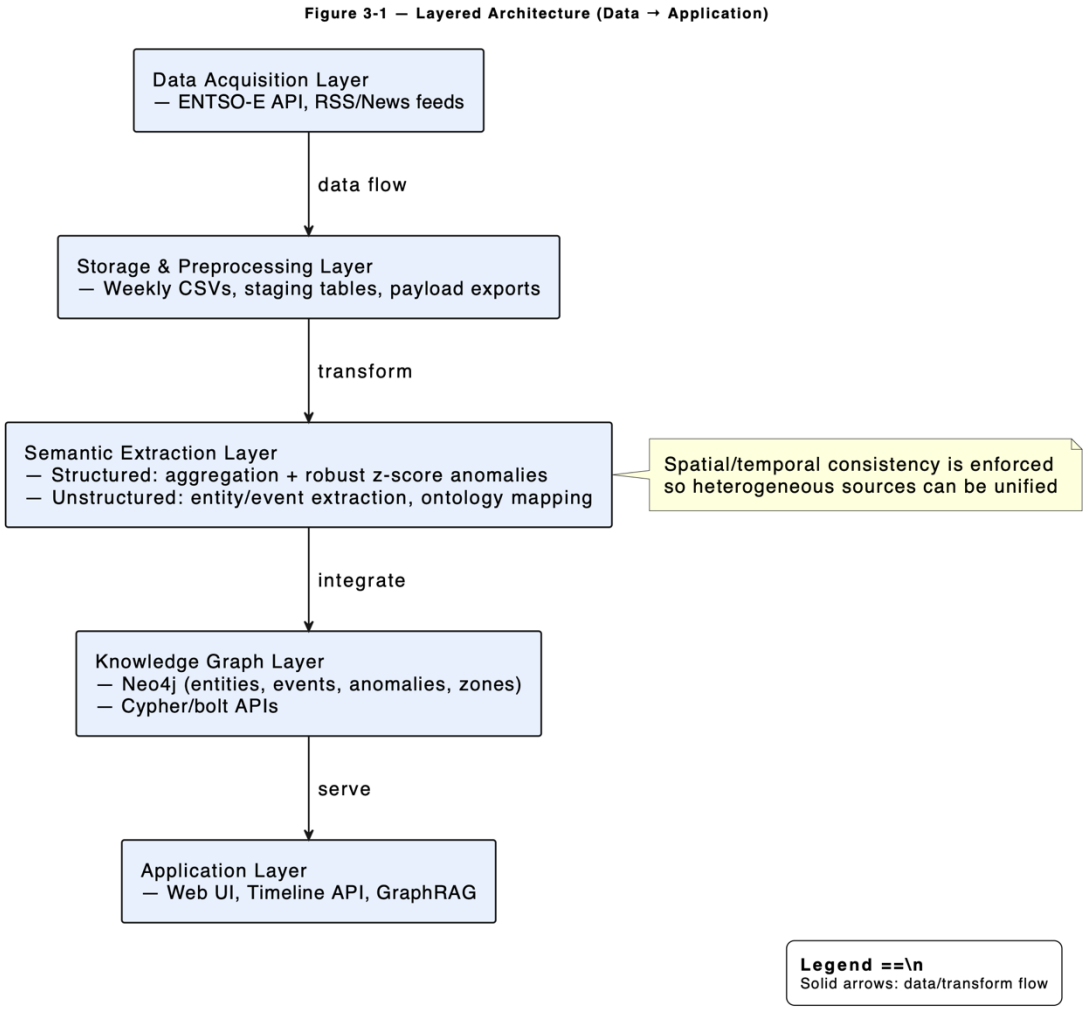


Figure 3-1: Layered architecture of the proposed system, illustrating the flow from data acquisition through preprocessing, semantic extraction, and integration into the knowledge graph, up to end-user applications.

3.2. Architecture Views

While the layered model provides a high-level static view of the system, it is also useful to consider the architecture from other perspectives. Figura 3-2 presents a data-flow and agent view of the system, highlighting how components interact dynamically. In this view, three primary workflows (or agents) handle different parts of the pipeline in sequence: a Structured Data ingestion workflow, an Unstructured Data extraction workflow, and a GraphRAG-based query workflow. Each workflow corresponds to one or more layers in the system and is implemented as an orchestrated set of tools and services. The structured workflow ingests time-

series data (e.g. CSV files of energy metrics) and detects anomalies, the unstructured workflow processes textual data (news articles) to extract semantic entities/events, and the GraphRAG workflow handles user queries by retrieving graph context and generating explanations. These components communicate via well-defined interfaces: for instance, the ingestion workflows read from files or APIs and write to the Neo4j graph database, while the GraphRAG agent reads from the graph (via Cypher queries) and interacts with a language model. The data flow between stages is also depicted – raw data (CSV files, JSON article dumps) flows into the system, is transformed into graph-structured knowledge, and finally is used to produce contextual explanations for end-user queries. This multi-view depiction underscores how the layers are realized in practice by specialized processes and how data moves through the system from inputs to insights.

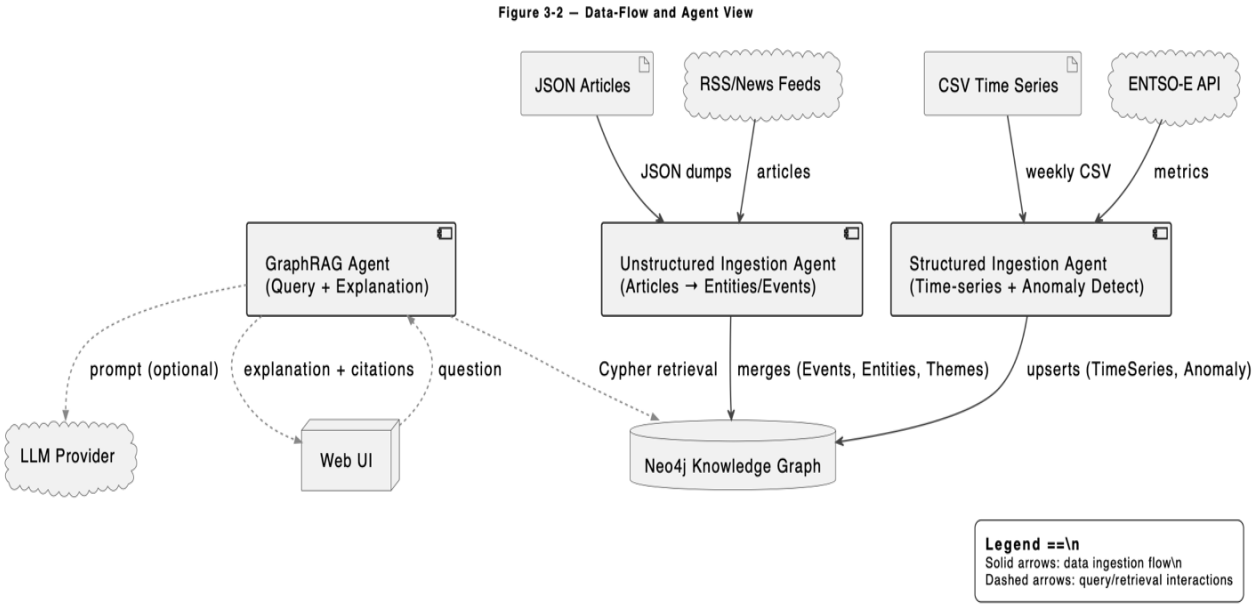


Figure 3-2: System architecture diagram showing data-flow and agent views. Structured and unstructured ingestion modules (agents) populate the Neo4j knowledge graph, and a GraphRAG query agent utilizes the graph to generate context-rich answers.

The remainder of this section shows how these layers are realized by deployable components and how they are deployed in the SaaS environment.

Component View

The component view illustrates the microservices and their interactions:

1. API Edge: a Gateway/BFF terminates HTTPS and routes requests to backend services.
2. Application Services: Timeline API returns anomaly timelines; Graph API serves graph queries; GraphRAG composes subgraph context with an LLM to produce explanations.
3. Ingestion Services: a Structured worker fetches ENTSO-E metrics, aggregates and detects anomalies; an Unstructured worker extracts entities/events from articles and maps them to the ontology.
4. Storage: Neo4j persists the integrated knowledge graph; optional CSV exports are produced for reproducibility.
5. External: LLM provider (optional for extraction and required for GraphRAG) and upstream data sources.

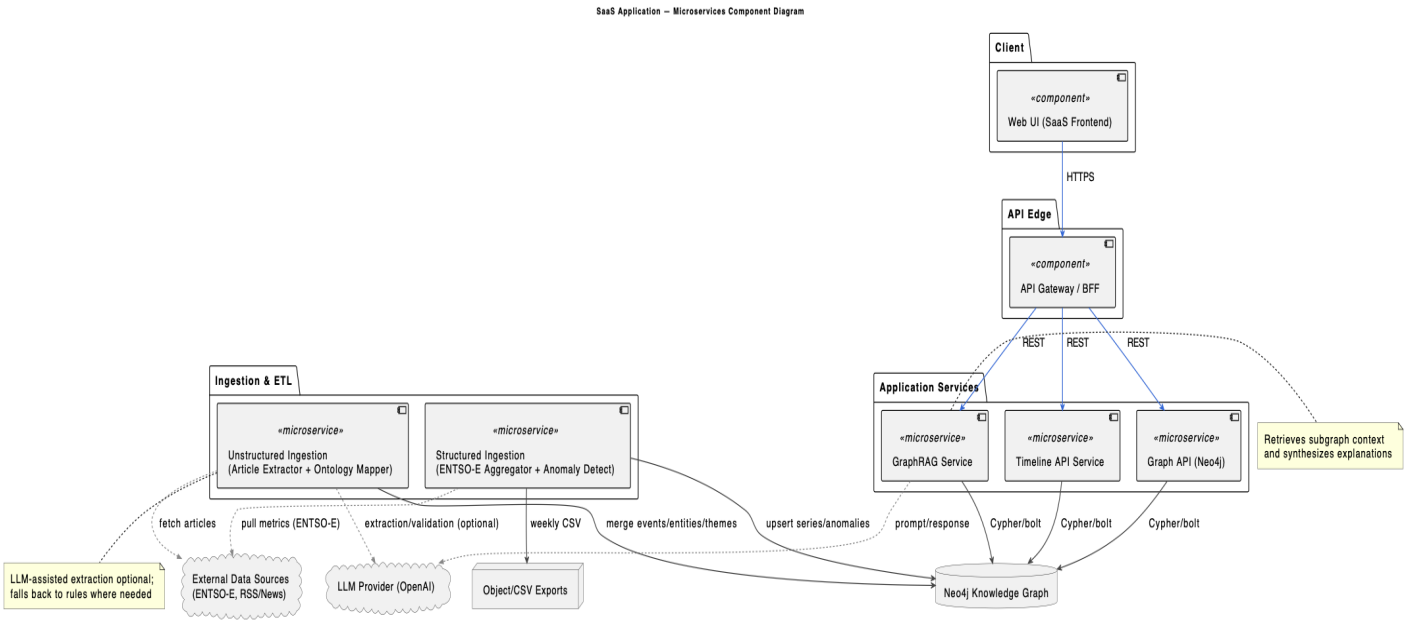


Figure 3-3: Microservices component diagram

Deployment View

The deployment view shows how the system is hosted:

1. User traffic reaches the cluster via an ingress/load balancer. The Web UI serves static assets; API Gateway handles REST endpoints.
2. Backend pods run independently and scale horizontally: Timeline API, Graph API, GraphRAG.
3. Ingestion pods run on schedules or event triggers, pulling from external sources and upserting into Neo4j.
4. Secrets/configs mount DB and LLM credentials; all service-to-DB traffic uses bolt/Cypher.

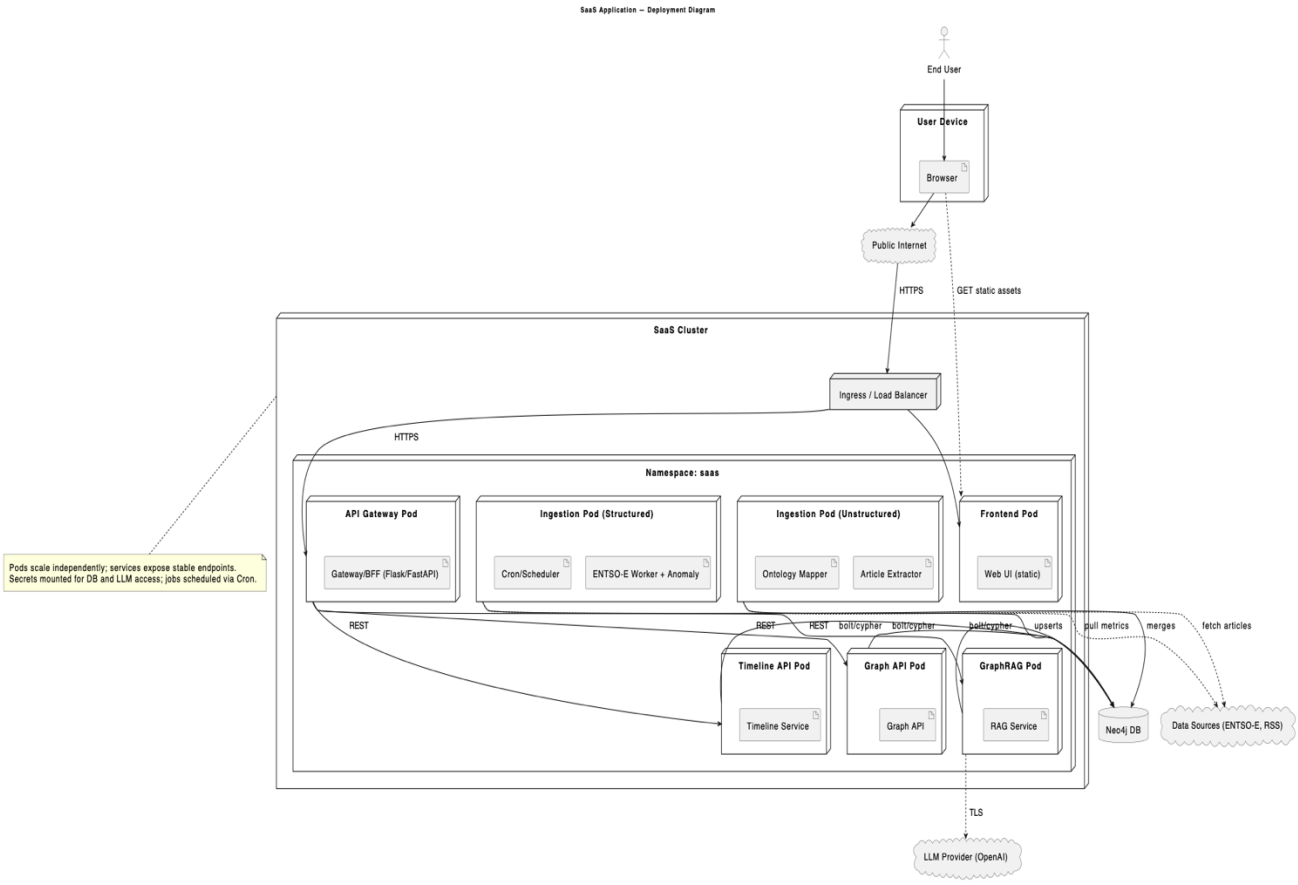


Figure 3-4: Deployment diagram

3.3. Data Sources & Contracts

The system integrates multiple open data sources, each with a defined input contract to ensure smooth ingestion and harmonization. Key sources include:

- **ENTSO-E Transparency Platform** – a European energy market data repository providing timeseries for electrical metrics (e.g. load demand, generation output, cross-border flows, day-ahead market prices). Data is obtained via the ENTSO-E API using an authentication token, or alternatively from published files. For consistency, raw hourly data is aggregated into weekly intervals in a standardized CSV format with common timestamp and value columns (e.g. _time in ISO 8601 UTC and value for the measurement). Each dataset (such as Greece electricity load 2015–2023) is thus converted into a CSV with a unified schema. The data contract here stipulates that timezones are converted to UTC and units are consistent across countries, so that all time-series share a common temporal frame and can be compared directly. If live API access is unavailable (e.g. missing token), the system can fall back to previously fetched CSV files, ensuring the pipeline remains offline-safe and reproducible.

- **Greek TSO & Grid Data (ADMIE)** – (Planned) open datasets from the Greek Independent Power Transmission Operator, which include electrical infrastructure information and possibly grid topology or regional load data. Although not heavily used in the current pipeline, the architecture is designed to accommodate such spatial infrastructure data. For example, grid node or region definitions from ADMIE can be ingested to enrich the knowledge graph's geospatial context (such as adding nodes for substations or transmission lines). These datasets typically come in tabular or geospatial formats; the system expects them to be converted into structured form aligned with the ontology (e.g. a substation would map to an Infrastructure or Facility entity). The contract for such data emphasizes consistent identifiers (e.g. using a known code for each region or asset) and coordinate information for mapping.

- **Meteorological and External Factors** – (Planned) Sources like weather data or market indicators can be integrated in the future. The architecture anticipates feeds such as temperature anomalies or fuel prices, which can be linked to energy events. Although not implemented in this work, any time-based external dataset can be incorporated by

defining a loader that produces a CSV or JSON with the required fields (time, location, value) and by extending the ontology with appropriate entity types (e.g. an ExtremeWeatherEvent). The design ensures new data types can be added with minimal changes, owing to the flexible schema of the knowledge graph (see Section 3.4).

- **Energy News Articles** – an extensive collection of textual data from Greek-language energy news outlets and RSS feeds. These unstructured sources provide contextual information on policy changes, market developments, and socio-economic events impacting the energy sector. Articles are collected via web scraping and news APIs and stored in a JSON repository (the "article warehouse") following a predefined Article schema. This schema defines fields such as id, title, author, content, published_date (in ISO 8601 format), language, sourceURL, etc., aligned with the Schema.org NewsArticle format. For instance, each article JSON includes a word count and a language code, and we enforce that published_date is in UTC. The ingestion contract for articles requires basic data cleaning: duplicate or near-duplicate articles (e.g. the same news from two sources) are identified (by identical title and date) and merged to avoid repetition, as per best practices ("remove duplicate or irrelevant observations"). We also filter out items that do not meet quality criteria (e.g. very short texts or non-Greek content), ensuring that only relevant, rich text is fed into the extraction pipeline. Crucially, each article record carries provenance metadata such as the original source URL and the retrieval timestamp. This provenance information ("the history of the state, custody, or location of something") is embedded so that any piece of extracted knowledge can be traced back to its source, supporting transparency and reproducibility.

3.4. Knowledge Graph Schema

All processed information converges into a central knowledge graph implemented as a Neo4j property graph. In this model, entities are represented as nodes with labels and properties, and relationships between entities are represented as typed edges. The graph's schema is designed to capture the key domain concepts of open energy data and their connections, in alignment with our ontology. Tables 3-1, 3-2 below summarize the main node types and relationship types in the knowledge graph:

Label	Purpose	Key Properties	Example Node
TimeSeries	Stream of ENTSO-E measurements (load, generation, price)	`product`, `unit`, `country`, `zone_code`, `period_start`, `period_end`	`TimeSeries (product='day_ahead_price')`
Anomaly	Detected outlier/shift on a time series	`start_ts`, `end_ts`, `score`, `classification`, `detected_by`, `pass`	`Anomaly(classification= 'LevelShift::Rise')`
Article	News/report document	`title`, `url`, `published_at`, `language`, `source`	`Article(publisher='Reuters')`
Chunk	Text span extracted from an article	`id`, `offset_start`, `offset_end`, `text`	`Chunk(id='article123::0-512')`
TemporalEvent	Event extracted from text	`event_type`, `title`, `start_ts`, `end_ts`, `time_text`	`TemporalEvent(event_type='PolicyChange')`
Entity	Actor/asset (organization, plant, regulator, person)	`name`, `class`, `country`, `entity_id`	`Entity(name='PPC / ΔΕΗ', class='Organisation')`
Theme	Abstract tag/category (Weather, Market, Regulatory, …)	`id`, `name`	`Theme(name='Weather')`
Zone	Bidding zone / geographic area	`code`, `name`, `canonical_name`, `country_code`	`Zone(code='10YGR-HTSO-----Y')`
Country	ISO3 country reference	`code`, `code_alpha2`, `name`	`Country(code='GRC', name='Greece')`

Table 3-1: Key node types in the knowledge graph (property graph model) and their roles.

Type	From → To	Purpose
HAS_ANOMALY	`TimeSeries` → `Anomaly`	Link measurements to detected anomalies
LOCATED_IN	`TimeSeries` → `Zone`	Anchor time series to bidding zones
HAS_CHUNK	`Article` → `Chunk`	Retain text spans supporting downstream retrieval
REPORTED_BY	`Chunk` → `Anomaly`	Tie anomalies to evidence passages
DESCRIBED_BY	`Article` → `TemporalEvent`	Attach structured events extracted from the article
MENTIONS	`Article` → `Entity`	Capture actors/assets appearing in the article
TAGGED_THEME	`Article` → `Theme`	Tag articles with topical themes
TAGGED_THEME (event)	`TemporalEvent` → `Theme`	Classify events by theme
TAGGED_THEME (entity)	`Entity` → `Theme`	Optional entity-level thematic tagging
INVOLVES	`TemporalEvent` → `Entity`	Participants/actors involved in the event
LOCATED_IN (event)	`TemporalEvent` → `Country`	Event geography (country-level anchoring)
REFERENCES_LOCATION	`Article` → `Zone`/`Country`	Geospatial references extracted from text

Table 3-2: Relationship types in the knowledge graph and their meaning.

Derived relationships (query-level, not persisted as edges)

- **NEAR_EVENT (Anomaly ↔ TemporalEvent):** temporal proximity (overlap or Δ± window) is computed in Cypher during retrieval; it is not stored as a permanent relationship.

- Event **LOCATED_IN:** optional; events can be grounded by associated entities/zones or by time-window and article evidence. The pipeline does not guarantee a persistent (TemporalEvent)-[:LOCATED_IN]->(Zone) edge.

As shown above, the property graph captures both structured data relationships (e.g. which time series generated, which anomalies, which region data belongs to) and contextual semantic relationships from unstructured data (e.g. which events are reported in which articles, which entities and locations those articles mention). It is a unified schema where, for instance, an Anomaly from a load time series can be connected (via NEAR_EVENT) to a TemporalEvent extracted from an article, which in turn might be linked to an Entity (like a company) and a Location (country). This enables complex queries such as, "Find all anomalies in country X that coincide with policy events involving company Y."

The graph schema was derived from an ontology of the energy domain (detailed in Section 3.5), but implemented in Neo4j's property graph model for pragmatic reasons. A property graph offers flexibility in schema evolution (new node or relationship types can be added without altering a rigid global schema) and efficient traversal for multi-hop queries. Traditional relational databases would struggle with this highly connected data, requiring complex JOIN operations across numerous tables (anomalies ↔ events ↔ entities ↔ locations) and a fixed schema that is hard to extend. In contrast, using a graph database has several benefits: it provides a natural representation of relationships (data is literally stored as nodes and edges), efficient graph traversals (finding all events related to a given anomaly is a matter of following relationships, not constructing JOINs), and a flexible schema that can accommodate evolving entity types. Furthermore, the graph format is amenable to the needs of retrieval-augmented generation (RAG): relevant subgraphs can be pulled as needed to supply factual context to an LLM, enabling explainable answers (Section 3.6).

We enforce data integrity in the graph through constraints and careful use of the Cypher query language. Wherever appropriate, unique identifiers or combinations of properties are defined for node types – for example, each Zone has a unique code, each TimeSeries can be uniquely identified by its country+product+zone, and each Article by a unique URL or title-timestamp combination. Neo4j unique constraints are applied on such fields to prevent duplicate nodes. Ingestion uses idempotent operations (Cypher MERGE) so that running the pipeline repeatedly will not create duplicate entries for the same real-world entity or event. These design choices ensure that the knowledge graph remains consistent even as new data is ingested incrementally or the extraction processes are re-run.

It is worth noting that the graph model also stores spatial and temporal attributes in a way that leverages the database's capabilities. Location nodes are enriched with coordinate data (latitude/longitude) when available, stored as Neo4j Point values, and temporal values use Neo4j's native date-time types. This allows the use of built-in spatial and time indexing and queries – e.g. finding the nearest events to a given location, or querying anomalies within a date range – directly via Cypher functions and indexes. By structuring the data as a property graph with these considerations, we obtain a powerful knowledge base that underpins the system's analytical and explainability features.

3.5. Knowledge Extraction & Ontology Mapping

A core challenge in building the above knowledge graph is extracting structured knowledge from raw data sources and mapping it to the ontology (the conceptual schema of our domain). We address this via two parallel extraction pipelines: one for structured time-series data and one for unstructured text. Both pipelines are aligned with a common ontology to ensure that their outputs (anomalies, events, entities, etc.) use consistent types and identifiers.

Structured data extraction (anomaly detection): The time-series pipeline takes the preprocessed numerical data (e.g. weekly aggregated energy metrics) and identifies noteworthy patterns or outliers that merit representation in the knowledge graph. Rather than storing every data point, we extract higher-level knowledge in the form of Anomalies. An anomaly here represents a significant deviation or change in the normal pattern of a metric – for example, a

spike (a sudden short-term surge or drop in value) or a level shift (a sustained step-change in the time series mean level). These anomalies are detected using statistical methods applied on each TimeSeries. When an anomaly is detected, it is characterized (type, time window, severity) and then mapped to an Anomaly node linked to the corresponding TimeSeries node. In essence, this pipeline semantically labels the raw numerical signals, turning them into discrete events (anomalies) that can be correlated with other data. The ontology defines what types of anomalies we recognize (currently focusing on level shifts, spikes, and variance changes, as these align with domain-relevant events like policy shifts or short-lived crises). By enforcing this classification, we ensure that detected anomalies are described in consistent terms (e.g., a "demand drop" anomaly might be categorized as a spike downward in load). This structured extraction is entirely automated and repeatable – every time new data is acquired or existing data is reprocessed, the anomaly detection step will yield the same kind of objects, making it a reliable bridge between raw numbers and knowledge graph entries.

Unstructured data extraction (entity/event/theme extraction): To extract knowledge from text (news articles), we employ a pipeline built around a Large Language Model (LLM) that is guided by our domain ontology. The system uses an LLM-powered agent to perform intent recognition, named entity recognition (NER), and event extraction on each article. The extraction process is ontology-aligned, meaning the LLM is instructed to output information following the classes and relationships defined in our schema. We achieve this by carefully crafting prompts that include guidelines and examples for the model, as well as by providing a JSON schema for each type of entity we want extracted. In practice, for each article, the system generates a structured prompt along the lines of: "Extract all relevant information about energy events and entities from the following text. Output the results in JSON format with fields X, Y, Z as defined in the provided schema." We define separate JSON schemas for different ontology categories (e.g., one for TemporalEvent, one for Location, one for Actor/Organization, etc.), and the prompt may instruct the LLM to fill multiple schemas depending on content. By giving the model an explicit schema to follow, we significantly improve the semantic fit and reliability of the output – the LLM is effectively constrained to produce well-formed JSON that matches expected types. Prior work and our observations confirm that providing a JSON Schema as guidance helps the LLM to output structured data that is immediately usable by downstream systems.

The LLM extraction workflow can be summarized in steps: (1) Load an article from the article warehouse, (2) construct the prompt with appropriate schema instructions (and occasionally few-shot examples for tricky entity types), (3) invoke the LLM to get a response, (4) parse the returned JSON, and (5) validate it against the schema for each ontology class. If the JSON passes validation, the extracted entities/events are accepted; if not (e.g., the LLM output is missing a required field or includes an unknown category), that article's result is put into an "unmapped" queue for manual review. This review mechanism acts as a safety net to ensure that we do not insert low-quality or semantically out-of-scope data into the knowledge graph. It also provides feedback for improving prompts or the ontology in the future (for instance, if many articles produce unmapped output, the ontology might need extension).

Through this pipeline, unstructured text is transformed into instances of our ontology. A sentence like "On 15 March 2022, the Greek regulator announced an emergency price cap." would result in a TemporalEvent node (with properties: title="Emergency price cap announced", start_ts≈2022-03-15) and possibly an Entity node (name="Greek energy regulator", type=ORGANIZATION) and a link between them (Entity is an actor in that event), all derived from one article. The system's ontology covers a comprehensive set of classes to capture such information – including temporal events, organizations (market actors, regulators, companies), infrastructure elements, socio-economic indicators, and so on (as introduced in Chapter 2). Each extracted item is mapped to one of these ontology classes. For example, an organization like ΑΔΜΗΕ ("ADMIE") in text is recognized by the LLM agent and mapped to the ontology class TransmissionOperator (a subtype of Entity), ensuring that the node we create in the graph is typed appropriately (instead of just a generic label). This ontology-driven approach curtails ambiguity (we know what kind of thing each name refers to) and consolidates synonyms or aliases (e.g. "ΔΕΗ" and "Public Power Corporation" would map to the same entity class if referring to the same organization).

The ontology mapping is also responsible for canonicalizing references, especially for spatial and temporal information. We instruct the LLM (and use post-processing) to normalize dates (e.g. convert "March 15, 2022" to 2022-03-15 ISO format) and to prefer standardized location names or codes. For instance, if an article mentions "Hellenic Republic" or "Greece", the system will recognize it as the country Greece and use a consistent name or country code

"GR" across the graph. In cases where the LLM's output for locations is not already canonical, a subsequent location normalization agent or lookup is employed (this is discussed in Stage 5 of the pipeline, Section 3.8). Thus, ontology mapping encompasses not just classifying extracted items but also aligning them to canonical identifiers (for locations, organizations, etc.), which is crucial for merging information from multiple sources about the same real-world entity.

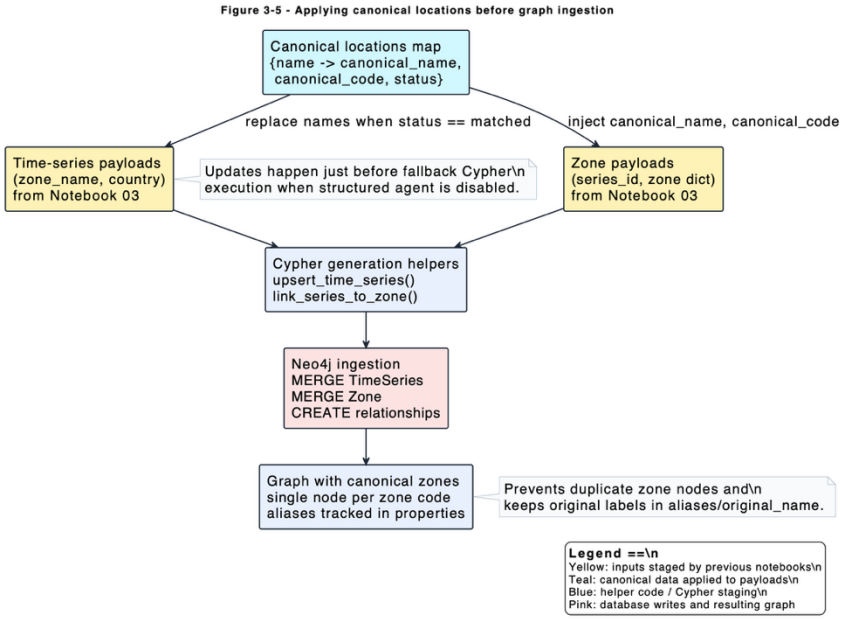


Figure 3-5: Applying canonical locations before graph ingestion

All extracted data, once validated and mapped, is inserted into the Neo4j graph via Cypher queries. The insertion logic uses MERGE operations (to avoid duplicates) and attaches appropriate relationships. For example, if an article extract yields an event and an organization, the pipeline will MERGE the TemporalEvent and Entity nodes (creating new ones if they don't exist already) and then create an MENTIONS or relevant relationship from the Article node to the Entity, and possibly between Event and Entity (depending on the ontology, e.g. Actor relationships). Each inserted node or relationship carries a reference back to the source (via properties or an attached source attribute) so that provenance is preserved within the graph as well.

3.6. GraphRAG Explainability

Our explainability layer follows a GraphRAG (graph-based retrieval-augmented generation) approach that grounds answers in the knowledge graph and (optionally) leverages an LLM for narrative composition[2],[6],[7],[8]. When a user asks, for example, "Why did Greek day-ahead electricity prices spike in March 2022?", the system first executes Cypher queries over the graph to assemble relevant facts and only then composes a natural-language answer.

Cypher-first retrieval: Retrieval begins with anomaly anchors and explicit constraints (time, geography, ontology). The system queries:

- the anomaly window (start_ts, end_ts) for the relevant TimeSeries and Zone,
- TemporalEvents that occur within a window [start_ts − Δ, end_ts + Δ],
- Entities mentioned in related articles, and
- Themes tagged on those events.

Temporal proximity (anomaly ↔ events) is computed at query time via time-window overlap; we do not persist a NEAR_EVENT edge. Articles are connected to events via (Article)-[:DESCRIBED_BY]->(TemporalEvent), and spatial grounding uses (TimeSeries|Entity)-[:LOCATED_IN]->(Zone). This Cypher-first strategy narrows the information space with precision—spatio-temporal filters and ontology labels ensure we retrieve exactly what the question requires.

Hybrid graph + vector: For broad or fuzzy queries (e.g., "What major events affected the energy market in 2022?"), we can complement Cypher with semantic or full-text search over node text (event descriptions, article titles). This hybrid mode helps surface relevant nodes whose textual content matches concepts not explicitly modeled as structured properties. The graph remains the primary source of truth; vector/full-text indexes are an optional enrichment.

Grounded generation: Retrieved facts (anomalies, events with timestamps and themes, entities, citations/URLs) are converted into a structured prompt (or a directly rendered explanation when LLMs are disabled). The LLM's role—if enabled—is to weave these facts

into a coherent narrative while adhering to guardrails: cite sources, prefer "unknown" when evidence is insufficient, and avoid speculation. Because every claim in the answer references specific nodes/edges (and article URLs), explanations are verifiable and auditable.

Design alignment with implementation:

- Themes are attached to events: (TemporalEvent)-[:TAGGED_THEME]->(Theme).
- Articles describe events: (Article)-[:DESCRIBED_BY]->(TemporalEvent); entities are mentioned via (Article)-[:MENTIONS]->(Entity).
- Geography is canonicalized via Zone, linked from series/entities with LOCATED_IN. An explicit Event→Zone link is optional and not guaranteed.
- TimeSeries holds identifiers/metadata (id, product, unit, zone_code/name). Raw time-series arrays live in staged artifacts; anomalies are nodes with type=Rise/Drop, classification (e.g., LevelShift::Rise), start_ts/end_ts, score, detected_by, value.

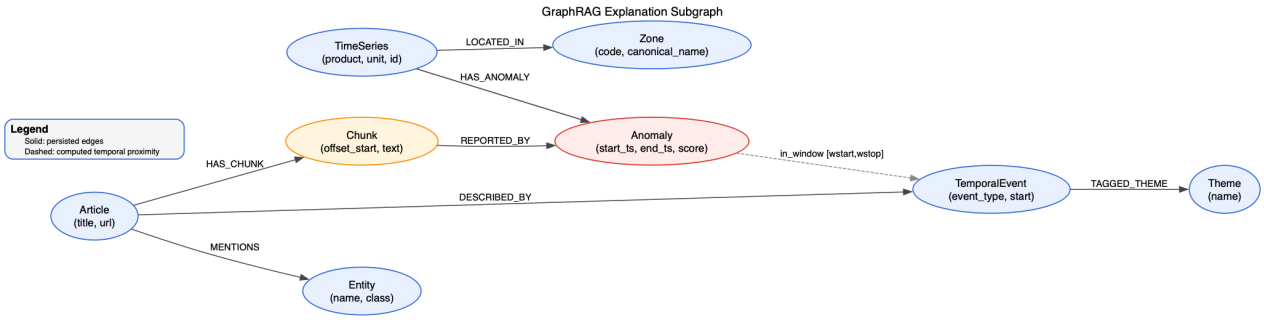


Figure 3-6: GraphRAG Explanation Subgraph

Retrieve events/entities/themes around an anomaly:

This design provides explainability-by-construction: the narrative is grounded in explicit graph facts with timestamps, zones, and sources; the graph is the source of truth; and (optional) LLMs act as readable narrators—not as unbounded information sources.

```
MATCH (ts:TimeSeries)-[:HAS_ANOMALY]->(an:Anomaly {id: $anomaly_id})
WITH an, an.start_ts AS start, an.end_ts AS stop
WITH an, start - duration({hours: $delta_h}) AS wstart, stop +
duration({hours: $delta_h}) AS wstop
OPTIONAL MATCH (art:Article)-[:DESCRIBED_BY]->(ev:TemporalEvent)
WHERE ev.start_ts >= wstart AND ev.start_ts <= wstop
OPTIONAL MATCH (art)-[:MENTIONS]->(e:Entity)
OPTIONAL MATCH (ev)-[:TAGGED_THEME]->(t:Theme)
RETURN an,
       collect(DISTINCT ev)[..$k] AS events,
       collect(DISTINCT e)[..$k] AS entities,
       collect(DISTINCT t)[..$k] AS themes;
```

Listing 3-1: Cypher query for anomaly-centric evidence retrieval

CHAPTER 4

4. IMPLEMENTATION

This chapter details the implementation of a pipeline for semantic integration of open energy data via knowledge graphs. The pipeline is implemented as a sequence of stages covering data preprocessing, anomaly detection, knowledge graph construction, unstructured data extraction, and explainable analysis via retrieval-augmented generation (RAG). We emphasize a Neo4j property graph as the core data model, using Cypher upsert patterns for idempotent ingestion and ensuring all components integrate seamlessly.

4.1. Structured Data Preprocessing (ENTSO-E)

The first stage involves collecting and preprocessing structured time series data from the ENTSO-E Transparency Platform, which provides open data on electricity generation, demand, and prices across Europe. We developed a data pipeline (Notebook 02) that fetches hourly ENTSO-E time series and converts them into weekly aggregates for downstream analysis. Figure 4.1 illustrates the data flow architecture for this stage. Key steps include data fetching via an API, optional staging in a time-series database, aggregation to weekly intervals, and export to CSV files:

- **Data Fetching:** We use the entsoe-py client to retrieve raw hourly time series (load, generation, day-ahead price, etc.) for each country/zone, given an API token. The data covers multiple years (e.g. 2019–2023) and is returned as Pandas DataFrames with timestamps and values.

- **InfluxDB Staging (Optional):** For performance, the pipeline can write the raw data to an InfluxDB time-series database if enabled (RUN_INFLUX_WRITE=1). This allows leveraging database queries for aggregation. If disabled, the pipeline proceeds with in-memory processing.

- **Timezone and Cleaning:** Timestamps are standardized to UTC and parsed with error coercion to handle any irregularities (e.g. missing or malformed timestamps). Each series is checked for completeness and cleaned of obvious errors (e.g. negative loads if not possible). All values are aligned to weekly boundaries (e.g. ISO weeks) for consistent aggregation.

- **Weekly Aggregation:** Hourly values (~8,760 points/year) are downsampled to weekly averages or sums (~52 points/year) to smooth short-term fluctuations and reduce data volume. This is done either via InfluxDB's query engine or with Pandas grouping, depending on the chosen path.

- **Export to CSV:** The resulting weekly time series are exported to a designated data directory as CSV files (one file per metric-country-year), using a standardized naming scheme {metric}_{country}_{year}_weekly.csv. Each CSV has two columns: _time (ISO-8601 timestamp for the week) and value (aggregated metric) for easy reuse in later stages.

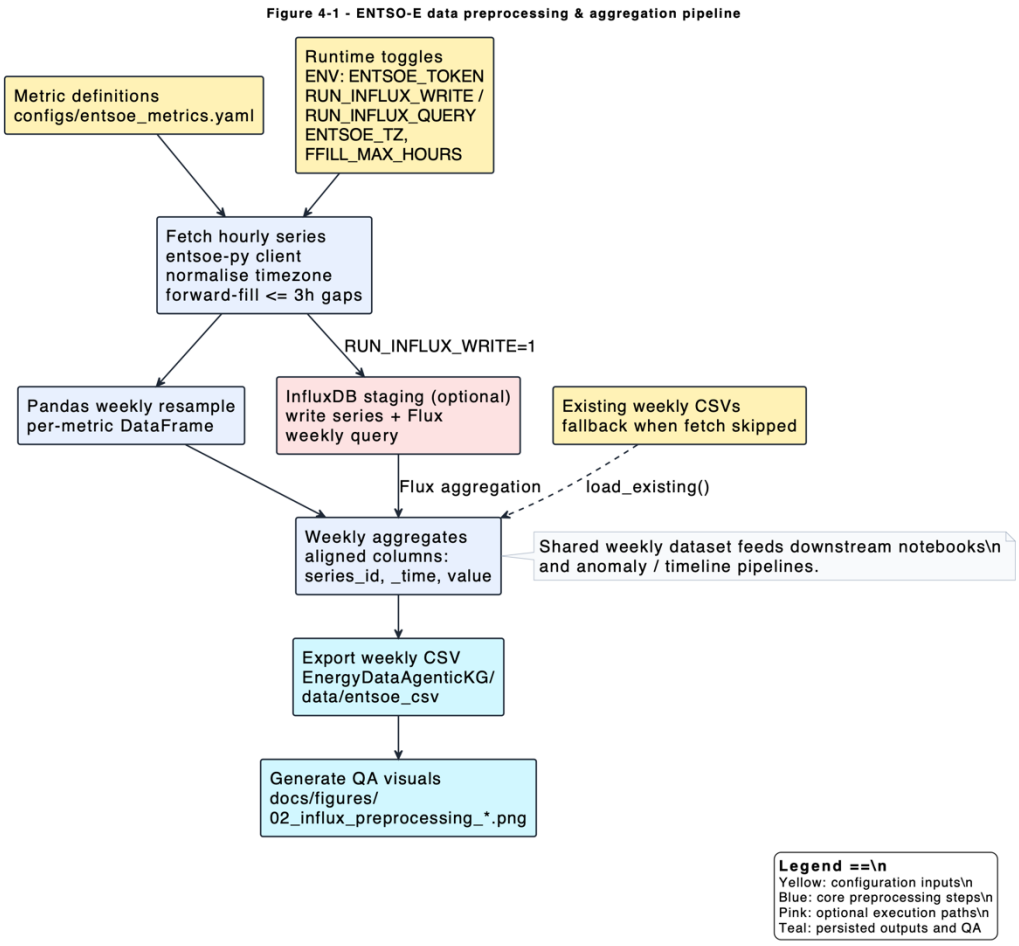


Figure 4-1: ENTSO-E data preprocessing and aggregation pipeline

To ensure reproducibility and safe re-running of this stage, the implementation checks for existing CSV files and will skip fetching if data is already available (preventing redundant API calls). All preprocessing steps are idempotent – running the pipeline twice produces the same outputs without duplication. Timezone normalization (coercing to UTC) and error handling guarantee consistency in the aggregated results. By the end of this stage, we have a set of weekly time series files per country and metric, ready for anomaly analysis.

4.2. Anomaly Detection and Structured Ingestion

After obtaining the weekly time series, the next step is to detect anomalies in these structured data and ingest both the time series and anomalies into the knowledge graph. Notebook 03 implements this structured data ingestion pipeline. It reads the weekly CSVs, applies statistical

anomaly detection algorithms, and then creates corresponding nodes and relationships in Neo4j[9],[10],[11]. The entire process transforms raw time series points into a graph representation linking each series with any detected anomalies in that series. Figure 4.2 depicts the anomaly detection and ingestion workflow.

Loading and Normalization: The pipeline begins by loading the weekly CSV files into Pandas DataFrames. Timestamps are parsed and normalized, and data is filtered to the target geographical region (for our case study, Greece and its bidding zone). We ensure each time series is labeled with metadata (country code GR, product type such as load or day_ahead_price, etc.) for identification. Basic statistical normalization may be applied if needed (e.g., scaling or detrending) to prepare data for anomaly detection.

Interactive Visualization: Before formal analysis, we incorporated an interactive visualization step using Plotly (or Matplotlib for static plots) to explore each time series. This helped in identifying seasonal patterns, trends, and potential outliers by visual inspection. Anomalies often manifest as sudden spikes, level shifts, or variance changes, which can be observed when plotting weekly values over multiple years. The interactive charts overlay the raw data with markers for any algorithm-detected anomalies, providing immediate visual validation of those detections.

Statistical Anomaly Detection: We implemented a combination of statistical methods to capture different types of anomalies in the time series. Specifically, the pipeline looks for: - Level shifts: sustained changes in the mean level of the series. We apply change point detection algorithms (like PELT or CUSUM) to identify significant shifts in the mean level. - Spikes: acute single-point outliers or short-lived extreme changes. For these, a robust z-score method is used, which computes the deviation of each point from a rolling median in units of robust standard deviation (e.g., median absolute deviation). Points beyond a threshold (e.g., $|z| > 3$) are flagged as spike anomalies. This method is less sensitive to long-term trends and focuses on local aberrations. - Variance changes: shifts in volatility, detected via moving window variance comparisons or statistical tests (e.g., Brown–Forsythe test). A significant change in variance within a sliding window indicates a regime change in volatility (which might correspond to regulatory changes or market shocks).

Each detected anomaly is classified by type (e.g., LevelShift, Spike, VarianceChange) and assigned a severity score. The robust z-score approach, for instance, yields a z-value as an anomaly score for spikes, while level shift algorithms provide confidence or cost metrics that we normalize to a 0–1 severity scale.

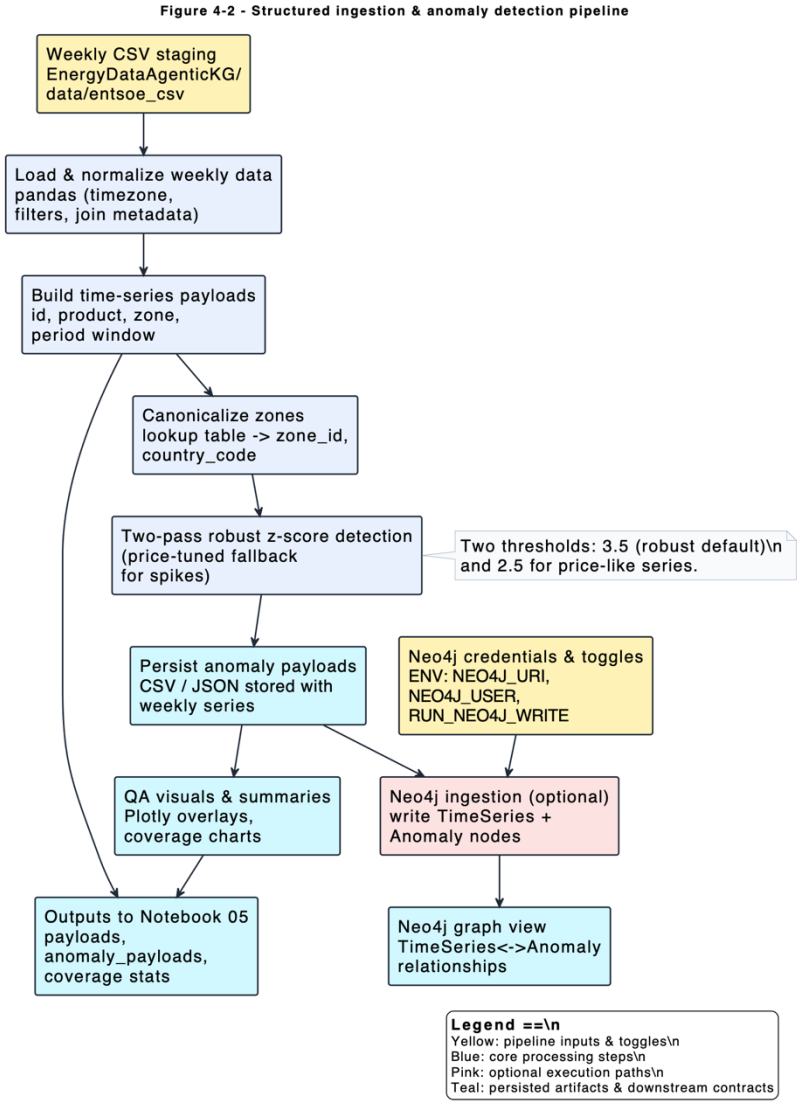


Figure 4-2: Anomaly detection and structured data ingestion pipeline. Weekly time series are read from CSV, anomalies are detected by statistical methods, and results are ingested into the graph as nodes and relationships.

Graph Schema for Structured Data: In this stage, two primary node types are created in Neo4j: - TimeSeries nodes: representing each unique time series (e.g., "Greece – load demand"). Key properties include country (ISO country code, e.g., "GR"), product (the metric

type, e.g., "load" or "day_ahead_price"), zone (bidding zone identifier, if applicable), and a list of data_points (the time series values or a reference to the CSV data). TimeSeries nodes effectively store metadata about the series and can optionally include the actual series data in compressed form (e.g., as a JSON array of weekly values). - Anomaly nodes: representing each detected anomaly instance. Properties include type (categorical label like "LevelShift", "Spike"), start_ts and end_ts (timestamps delimiting the anomaly period; for a spike these might be identical or a single week; for a level shift it could span multiple weeks), and a numeric score or severity indicating confidence or magnitude of the anomaly.

We define a relationship HAS_ANOMALY from TimeSeries to Anomaly to link each time series with the anomalies found in it. Each Anomaly node is thus attached to the specific TimeSeries it came from, and can later be related to external events or explanations. Additionally, we capture geographic context by linking each TimeSeries to a Zone (or Location) node representing the grid zone or country (via a :LOCATED_IN relationship). For example, a time series node for Greek data might have (:TimeSeries)...-[:LOCATED_IN]->(:Zone {code:"GR"}). Zones and locations are discussed more in Section 4.4, but note that using a consistent zone/country node enables connecting anomalies with external data by geography.

After ingestion, validation queries are executed to ensure data quality. For instance, we run Cypher queries to count the number of anomalies per series, check that each anomaly has the expected properties, and verify that unique constraints are not violated. This stage establishes the foundation of the knowledge graph: a set of time series with their anomalies, anchored by location nodes. The pipeline is designed so that re-running it will update existing nodes rather than duplicate them (achieved through MERGE operations for upserts, see Section 4.4), ensuring idempotency.

4.3. Unstructured Data Extraction (Articles to Events)

While structured ENTSO-E data provides quantitative anomalies, explaining those anomalies requires qualitative context (events, news, policy changes). The third stage of the pipeline focuses on unstructured data ingestion: processing text from news articles and reports to extract key events and entities related to energy markets. We leverage a Large Language Model (LLM) powered agent (via Google's ADK – Agent Development Kit) to perform ontology-guided

information extraction from text. Notebook 04 implements this stage, turning raw text articles into structured graph data.

Article Collection: We assume a corpus of energy-related news articles has been collected (e.g., via RSS feeds or web scraping) and stored as JSON files. Each article record contains fields like title, content, publication date, source, etc. In our case, articles from 2019–2023 covering the Greek energy market were gathered from sources such as newswire services (Reuters, Bloomberg), energy agencies (ENTSO-E news, IEA reports), and local press. We load these articles from the data/articles/ directory into memory for processing.

Relevance Filtering: A pre-processing filter discards articles that are not relevant to our domain or target language. We use language detection to keep only English-language articles (since our extraction model is configured for English). We also apply keyword filtering: articles must contain energy-related terms (e.g., "electricity", "power", "energy", "gas", "grid") to be processed. This filtering ensures we focus the expensive LLM extraction on pertinent articles. In practice, about a few hundred articles remain after filtering for the given time frame and domain.

LLM-Powered Information Extraction: For each relevant article, we employ a custom ADK agent to extract structured information. The agent is prompted with instructions to identify and extract: - Temporal events: Significant events mentioned in the article (e.g., policy changes, infrastructure outages, market disturbances), including when they happened. - Entities: Key actors or organizations (e.g., energy companies, government bodies, persons) mentioned. - Themes or topics: Major themes or categories (e.g., "market regulation", "renewable integration", "energy crisis"). - Locations: Any geographic references (countries, regions, facilities) involved.

The agent prompt is guided by an ontology schema we defined for the energy domain, ensuring the output follows a JSON structure with specific fields for each category. Listing 4.1 shows an excerpt of the structured output schema expected from the LLM agent, which aligns with our ontology:

```
{
  "events": [                          // Temporal events from the
article
    {
      "title": str,                    // Event name or summary
      "start_date": str,               // ISO 8601 date or datetime of
event start
      "end_date": str,                 // (Optional) ISO 8601 end
date, if applicable
      "description": str,              // Detailed description of the
event
      "classification": str,           // Event type (e.g., policy,
market, technical)
      "location": str                  // Named location reference (as
mentioned in text)
    }
  ],
  "entities": [                        // Named entities
(organizations, people, etc.)
    {
      "name": str,                     // Entity name
      "type": str,                     // Type of entity
(ORGANIZATION, PERSON, etc.)
      "relevance": float               // Relevance score (0.0–1.0) or
confidence
    }
  ],
    {
      "label": str,                  // Theme label
      "category": str                // Broad category of the theme
    }
```

Listing 4-1: JSON schema for LLM-based article extraction (ontology-guided). The agent outputs a structured JSON with events, entities, themes, and locations identified in each article.

```
  "themes": [                          // Thematic tags or categories
    {
      "label": str,                    // Theme label
      "category": str                  // Broad category of the theme
    }
  ],
  "locations": [                       // Geographic references
    {
      "name": str,                     // Location name as mentioned
(e.g., "Athens")
      "country_code": str,             // ISO country code if
identifiable (e.g., "GR")
      "geo_type": str                 // Granularity: country, city,
region, etc.
    }
  ]
}
```

Listing 4-2: JSON schema for LLM-based article extraction (ontology-guided). The agent outputs a structured JSON with events, entities, themes, and locations identified in each article.

The agent uses a GPT-4 model (via the ADK's LiteLLM interface) to generate this JSON output for each article, following the above schema. By providing a precise schema and examples in the prompt (few-shot learning), we guide the LLM to output parseable JSON. This dramatically reduces hallucinations and ensures that the extraction conforms to our knowledge graph ontology (each key corresponds to a node type we will store). The use of a structured output prompt is a Cypher-first approach in spirit: rather than free-form text, the LLM is tasked with producing data that can be directly ingested into the graph.

Parsing and Post-processing: The JSON output from the LLM is parsed by our pipeline into Python objects. We then perform minor post-processing: - Normalize date strings to consistent datetime formats (e.g., ensure start_date and end_date are in ISO8601 and convert to UTC timezone). - Discard or merge low-relevance entities (e.g., if the agent extracted many entities with low relevance scores, we might keep only those above a threshold or aggregate duplicates). - Preliminary canonicalization of names (for locations and organizations) if straightforward (e.g., converting "U.S." to "United States"). More advanced canonicalization is done in the next stage.

Graph Ingestion of Unstructured Data: For each article processed, we create the following in Neo4j: - An Article node, with properties: title, published_at (timestamp), source (e.g., "Reuters"), and potentially storing the text content or a reference to it. - One or more TemporalEvent nodes for each event extracted from the article. Each TemporalEvent has title (a short name), start_ts, end_ts, description, and classification (a label like "Policy" or "MarketEvent"). - Entity nodes for organizations or people (if not already present in the graph). Each has name and type (plus a relevance or significance score). - Theme nodes for thematic topics (e.g., "Energy Crisis" with category "Market Disruption"). - Location nodes for any locations mentioned (though for countries/zones we will later merge with existing nodes).

These nodes are connected with relationships to preserve provenance and context: - (:Article)-[:DESCRIBED_BY]->(:TemporalEvent) links an article to events it describes. - (:Article)-[:MENTIONS]->(:Entity) links to entities mentioned. - (:Article)-[:HAS_THEME]->(:Theme) links to identified themes/topics. - (:Article)-[:REFERENCES]->(:Location) links to location mentions in the text. - Additionally, we link each TemporalEvent to a Location if a specific location is associated (e.g., an event happening in Athens would get (:TemporalEvent)-[:LOCATED_IN]->(:Location {name:"Athens"})).

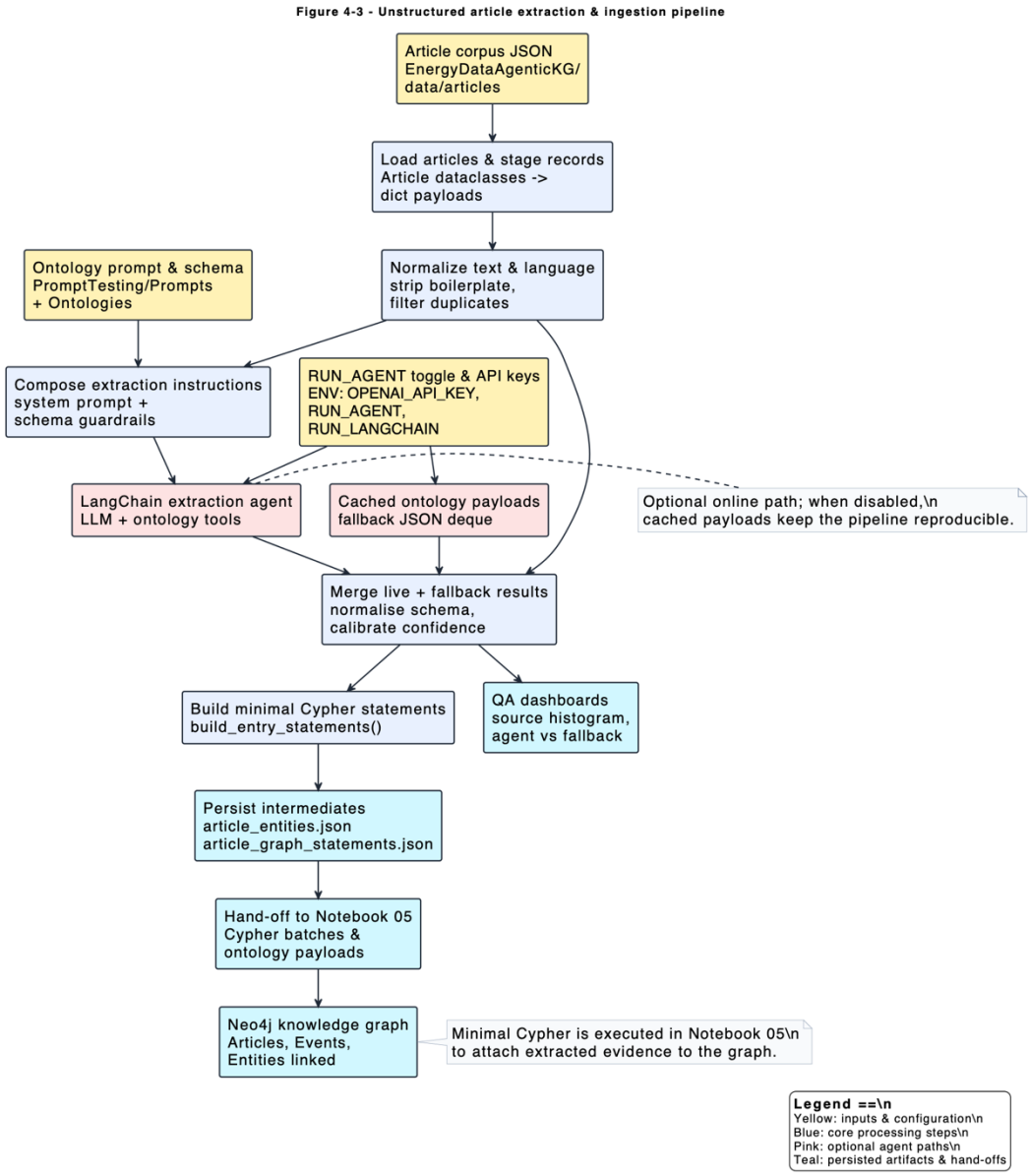


Figure 4-3: Unstructured data extraction and ingestion. Relevant news articles are processed by an LLM agent to extract events and entities, which are then added to the knowledge graph.

This stage results in a subgraph of articles, events, entities, themes, and locations, all interconnected and ready to be linked to the anomaly graph built earlier. By the end of Notebook 04, we have two largely separate subgraphs in Neo4j: one from structured ENTSO-E data (time series and anomalies), and one from unstructured text (articles and related entities). In the next section, we integrate these subgraphs through canonicalization and linking, creating a unified knowledge graph.

Prompt Policy & Ontology Schema

This subsection examines the prompt policy and ontology schema as implemented in the repository and explains how they are used by the extraction pipeline.
Below is code snippet screenshot depicting the architecture of the prompt-system:

Compose System Prompt from Ontology Components

Purpose: Merge natural language prompt and formal schema into a unified LLM instruction.

System Prompt Architecture

The system prompt serves as the **complete specification** that guides the LLM's extraction behavior. It combines three elements:

1. Role Definition (Establishes agent identity):

```
"You are a precise information extraction assistant.
 Follow the ontology prompt and schema strictly.
 Return VALID JSON. Explain any missing information
 inside a top-level 'reasoning' field."
```

Purpose: Sets expectations for:

- **Precision**: Extract exactly what's in the text (no hallucination)
- **Compliance**: Follow ontology structure strictly
- **Format**: Output must be parseable JSON
- **Transparency**: Document uncertainty in `reasoning` field

2. Ontology Prompt (Natural language guidance):

- Task description and goals
- Entity type definitions with examples
- Classification taxonomies
- Output format requirements
- Edge case handling instructions

3. Ontology Schema (Formal specification):

- JSON Schema with field definitions
- Required vs optional properties
- Data type constraints
- Validation rules
- Enumeration values for classifications

Why This Three-Part Structure?

Role Definition Alone → LLM understands general task but lacks domain knowledge

- Example: "Extract entities" → finds names but misses energy-specific context

Role + Prompt → LLM understands domain but output structure varies

- Example: Might extract "Greek power plant" but format inconsistently

Role + Prompt + Schema → LLM produces consistent, validated, domain-aware output

- Example: Extracts `{"name": "Greek power plant", "type": "INFRASTRUCTURE", "relevance": 0.85}`

Prompt Composition Function

```python
def compose_system_prompt(prompt_cfg: dict, schema_cfg: dict) -> str:
    return (
        # Part 1: Role definition
        "You are a precise information extraction assistant. "
        "Follow the ontology prompt and schema strictly.\n"
        "Return VALID JSON. Explain any missing information "
        "inside a top-level 'reasoning' field.\n\n"

        # Part 2: Natural language instructions
        "=== ONTOLOGY PROMPT ===\n"
        f"{json.dumps(prompt_cfg, ensure_ascii=False, indent=2)}\n\n"

        # Part 3: Formal schema
        "=== ONTOLOGY SCHEMA ===\n"
        f"{json.dumps(schema_cfg, ensure_ascii=False, indent=2)}\n"
    )
```

Listing 4-3: Prompt System Architecture

Listing 4-1: Prompt System Architecture

Prompt policy (ontology_prompt2.json): The prompt file specifies how the model should convert article text into schema-conformant JSON. Its structure encodes (i) a task instruction that forbids free-form output, (ii) a compact exposition of allowed classes and fields, and (iii) guardrails that control ambiguity and provenance.

- **Semantic-fit policy:** The prompt includes an explicit requirement to avoid forced mappings; when a fact does not clearly fit a class, it must be marked as "unmapped" (with a proposed class and justification) rather than coerced. This pattern is central to preserving ontology integrity during extraction, and it integrates directly with validation and unmapped queues downstream.

- **Temporal anchoring:** The prompt directs the model to normalize all instants/intervals to ISO 8601, resolving relative expressions (e.g., "last week") into explicit start_ts/end_ts when possible. This is critical for aligning event windows with anomaly windows during Cypher-first retrieval.

- **Bilingual labels:** Where applicable, labels should include both English and Greek variants, with a fallback to generic type labels when one language is missing. This supports localized UI rendering without sacrificing cross-language retrieval.

- **Themes on events:** The policy follows the graph design by placing themes on TemporalEvent (not on Article), to keep topical context anchored to time-stamped objects.

- **Provenance:** The policy requires article identifiers, URLs, titles, and published_at, as well as extraction_time, model, and prompt/schema version fields for auditability.

Ontology Prompt Structure

The prompt instructs the LLM on extraction tasks with specific guidance:

Task Definition:

```
{
  "task": "Extract structured information from energy sector news articles",
  "goal": "Identify entities, events, themes, and locations for knowledge graph construction",
  "output_format": "Valid JSON matching the provided schema"
}
```

Entity Type Definitions (examples):

```
{
  "entities": [
    {
      "type": "ORGANIZATION",
      "description": "Companies, agencies, grid operators, utilities",
      "examples": ["ENTSO-E", "RTE", "Public Power Corporation", "European Commission"],
      "attributes": ["name", "type", "relevance_score"]
    },
    {
      "type": "LOCATION",
      "description": "Countries, regions, cities, bidding zones",
      "examples": ["Greece", "GR", "Northern Italy", "IT_NORD", "Athens"],
      "attributes": ["name", "country_code", "geo_type"]
    }
  ]
}
```

Temporal Event Structure:

```
{
  "events": {
    "description": "Time-bound occurrences mentioned in the article",
    "required_fields": ["title", "start_date", "description"],
    "optional_fields": ["end_date", "classification", "location"],
    "classifications": [
      "policy_change",      // New regulations, laws, directives
      "market_event",       // Price changes, capacity additions
      "technical_issue",    // Outages, failures, maintenance
      "interconnection",    // Cross-border links, grid expansions
      "capacity_change"     // Generation additions/retirements
    ]
  }
}
```

Theme Categories:

```
{
  "themes": {
    "description": "High-level topics and subject classifications",
    "categories": [
      "Energy Crisis",
      "Renewable Energy",
      "Grid Infrastructure",
      "Energy Policy",
      "Market Dynamics",
       "Supply Security"
    ]
  }
}
```

Output Format Instructions:

```
{
  "format_requirements": {
     "dates": "ISO8601 format (YYYY-MM-DDTHH:MM:SSZ)",
    "json_validity": "Must parse without errors",
    "required_sections": ["entities", "events", "themes", "locations"],
    "error_handling": "If uncertain, include 'reasoning' field with explanation"
```

Listing 4-4: Ontology Prompt Structure

Ontology schema (ontology_schema_v11.json): The schema enumerates classes, properties, datatypes, and basic constraints. It is minimal by design but should be strict where correctness matters and flexible where ambiguity is expected.

- **Classes in scope:** Article, TemporalEvent, Entity, Theme, Zone (canonical geography), and (from structured ingestion) TimeSeries and Anomaly.

- **Required fields:** Article: id/url/published_at, TemporalEvent: id/event_type/start_ts (end_ts optional), Entity: id/entity_type/(label_en or label_el at minimum), Theme: id/name, Zone: code/name/type, Anomaly: id/series_id/start_ts/end_ts/type/classification/score/detected_by/value. TimeSeries nodes in the graph are stitched from structured ingestion, not from the unstructured pipeline.

- **Enumerations:** Constrain **TemporalEvent:** event_type and **Entity:** entity_type to bounded enums reject values that drift beyond the declared set. This keeps retrieval filters predictable.

- **Pattern constraints:** Adopt patterns for ids (e.g., ^(art|ev|ent|th|zone)*[A-Za-z0-9*\-]+$) and ISO 8601 timestamps; add min/max length where appropriate (e.g., title length).

- **Bilingual labels:** For any "label_*" field, enforce minCount 1 across the pair (label_en OR label_el) and allow both when available.

Ontology Schema Specification

The JSON Schema enforces structural validation and data types:

Top-Level Schema:

```
{
  "type": "object",
  "required": ["entities", "events", "themes", "locations"],
  "properties": {
    "entities": {
      "type": "array",
      "items": {"$ref": "#/definitions/Entity"}
    },
    "events": {
      "type": "array",
      "items": {"$ref": "#/definitions/TemporalEvent"}
    },
    "themes": {
      "type": "array",
      "items": {"$ref": "#/definitions/Theme"}
    },
    "locations": {
      "type": "array",
      "items": {"$ref": "#/definitions/Location"}
    },
    "reasoning": {
      "type": "string",
      "description": "Optional explanation of extraction decisions"
    }
  }
}
```

Entity Definition:

```
{
  "definitions": {
    "Entity": {
      "type": "object",
      "required": ["name", "type"],
      "properties": {
        "name": {"type": "string", "minLength": 1},
        "type": {
          "type": "string",
          "enum": ["ORGANIZATION", "PERSON", "INFRASTRUCTURE", "POLICY"]
        },
        "relevance": {
          "type": "number",
          "minimum": 0.0,
          "maximum": 1.0,
          "description": "Entity importance to article content"
        }
      }
    }
  }
}
```

Listing 4-5: Ontology Schema Specification

Temporal Event Definition:

```
{
  "definitions": {
    "TemporalEvent": {
      "type": "object",
      "required": ["title", "start_date", "description"],
      "properties": {
        "title": {"type": "string", "maxLength": 200},
        "start_date": {
          "type": "string",
          "format": "date-time",
          "description": "ISO8601 timestamp"
        },
        "end_date": {
          "type": "string",
          "format": "date-time"
        },
        "description": {"type": "string", "maxLength": 1000},
        "classification": {
          "type": "string",
          "enum": ["policy_change", "market_event", "technical_issue",
                   "interconnection", "capacity_change", "other"]
        },
        "location": {"type": "string"}
      }
    }
  }
`
```

Listing 4-6: Ontology Schema Specification

Processing path (PromptTesting/processor1.py): The script orchestrates article extraction with the policy and schema above, and it implements strict validation and unmapped handling.

- **Prompt composition:** The script prepares a system instruction block (the ontology policy) and a user block (article text and key metadata). Keep the system role minimal and focused on constraints; avoid mixing long enumerations when the schema already constrains them.

- **Model call:** The script issues a chat-style request (e.g., to GPT-4o-mini) with low temperature and a bounded max_tokens, and captures the raw JSON string. Any extraneous commentary is grounds for rejection by the parser.

- **JSON validation:** The script validates the JSON object against ontology_schema_v11.json using jsonschema (or equivalent). Validation errors (type, missing required field, enum mismatch) cause the payload to be rejected or routed to "unmapped" review, depending on policy.

- **Unmapped queue:** If "unmapped_entities" or similar structures are returned, the script logs and optionally persists them for later analysis rather than forcing these entries into the graph.

- **Fallback path:** In offline or cost-constrained runs, the script loads a deterministic fallback dataset of validated payloads, ensuring downstream pipeline stages remain operable.

4.4. Knowledge Graph Integration and Canonicalisation

With structured and unstructured data ingested, the next implementation step is to integrate them into a unified knowledge graph. Notebook 05 orchestrates this process, ensuring that nodes referring to the same real-world entities are merged and creating relationships between anomalies and the events or factors that explain them. The integration focuses particularly on geographic and entity canonicalization and on establishing temporal links between anomalies and events.

Graph Assembly: First, we verify the Neo4j database is accessible and pre-create any needed indexes or constraints. For example, we set unique constraints on certain node properties (like TimeSeries(country, product) combination or Location(canonical_code)) to prevent duplicates. We then ingest any remaining structured data that might not have been loaded in previous notebooks (Notebook 05 is designed to be a one-stop graph builder, so it can ingest the time series and anomalies if Notebook 03 wasn't run separately). In practice, since we already did that, this step finds existing TimeSeries and Anomaly nodes.

Geographic Entity Canonicalization: One major challenge is that the same location or region may appear under different names in structured vs. unstructured data. For example, the country Greece might be referred to as "GR" (ISO code) in the ENTSO-E dataset, but as "Greece", "Hellenic Republic", or even in Greek language ("Ελλάδα") in news articles. Without reconciliation, our graph would have separate Location or Zone nodes for each variant, fragmenting the knowledge. To address this, we implement a canonicalization module for geographic names: - We designate an authoritative representation for each location (for countries, we use the ENTSO-E/ISO country code, e.g., "GR" for Greece, as the canonical code, along with a standard name "Greece"). - We merge or link all location nodes that refer to the same real entity. In Neo4j, we use idempotent Cypher queries with MERGE to either find existing nodes or create new ones with the canonical identifier. For example, we MERGE

(loc:Location {canonical_code: "GR"}) and then set its properties (canonical_name = "Greece", aliases = [...]) to include all known aliases. If a Location node from the article ingestion (e.g., with name "Hellenic Republic") doesn't yet have the canonical code, it gets attached or merged into the canonical node. This way, we avoid duplicate nodes for "Greece" [10]. - We developed a two-tier canonicalization approach: (a) an ADK LLM agent that can resolve tricky cases by context (e.g., understanding that "Northern Macedonia" likely refers to "North Macedonia (MK)" vs. the Greek region Macedonia) for high accuracy, and (b) a deterministic Python lookup using a curated alias table for common names. The pipeline decides which to use based on configuration flags. Listing 4.2 shows pseudo-code for this selection logic:

```
if RUN_ADK and OPENAI_API_KEY:
    use_adk_agent_canonicalization()   # GPT-4 powered, high
accuracy
else:
    use_python_fallback_canonicalization()   # Fast, deterministic
lookup
```

Listing 4-7: Pseudo-code for location canonicalization. A fallback lookup table is used by default, with an optional LLM agent for complex cases.

Using this approach, a variety of variants are normalized to a single Location node with a canonical_code. For example, the Location node for Greece ends up with canonical_code = "GR", canonical_name = "Greece", and aliases = ["Hellenic Republic", "Greek grid", "Ελλάδα", "GR"]. The result is that a query for Location {canonical_code:"GR"} will match any reference to Greece from any source. Figure 4.4 illustrates the effect of canonicalization on the graph structure for locations.

Without canonicalization, multiple disconnected nodes represent the same place; with canonicalization, they unify into one node with all aliases:

```
• Without unification:
  (:Location {name: "Greece"}) (from structured data)
  (:Location {name: "Hellenic Republic"}) (from article A)
  (:Location {name: "Greek grid"}) (from article B)
  (:Location {name: "GR"}) (from zone code)
  // Results in 4 separate nodes that should be one.
• With canonicalization:
  (:Location { canonical_code: "GR", canonical_name: "Greece",
    aliases: ["Hellenic Republic","Greek grid","GR"],
    original_names: ["Greece","Hellenic Republic","Greek
  grid","GR"] })
  // Results in a single node with unified identity and traceable
  provenance.
```

Listing 4-8: Canonicalization effect on data.

Once the canonical location nodes are in place, any duplicate location nodes are merged or related. In our implementation, if a duplicate was created (e.g., a Location from an article with name "Greece" without canonical_code), we update it to set canonical_code = "GR" and then use Neo4j's MERGE to ensure there is only one node with canonical_code "GR". The pipeline uses MERGE and ON CREATE/ON MATCH to attach new info without creating duplicates, and we rely on Neo4j unique constraints on Location(canonical_code) to guarantee a single node per code. This idempotent design allows running the integration step multiple times safely.

Entity and Theme Canonicalization: A similar approach is applied to other entities if needed. For instance, an organization might appear with slight naming differences (e.g., "European Commission" vs "EC"). In such cases, we either manually curate aliases or rely on string similarity to merge obvious duplicates. However, our main focus was on locations, since those are critical for linking structured and unstructured data (the ENTSO-E data is country-specific).

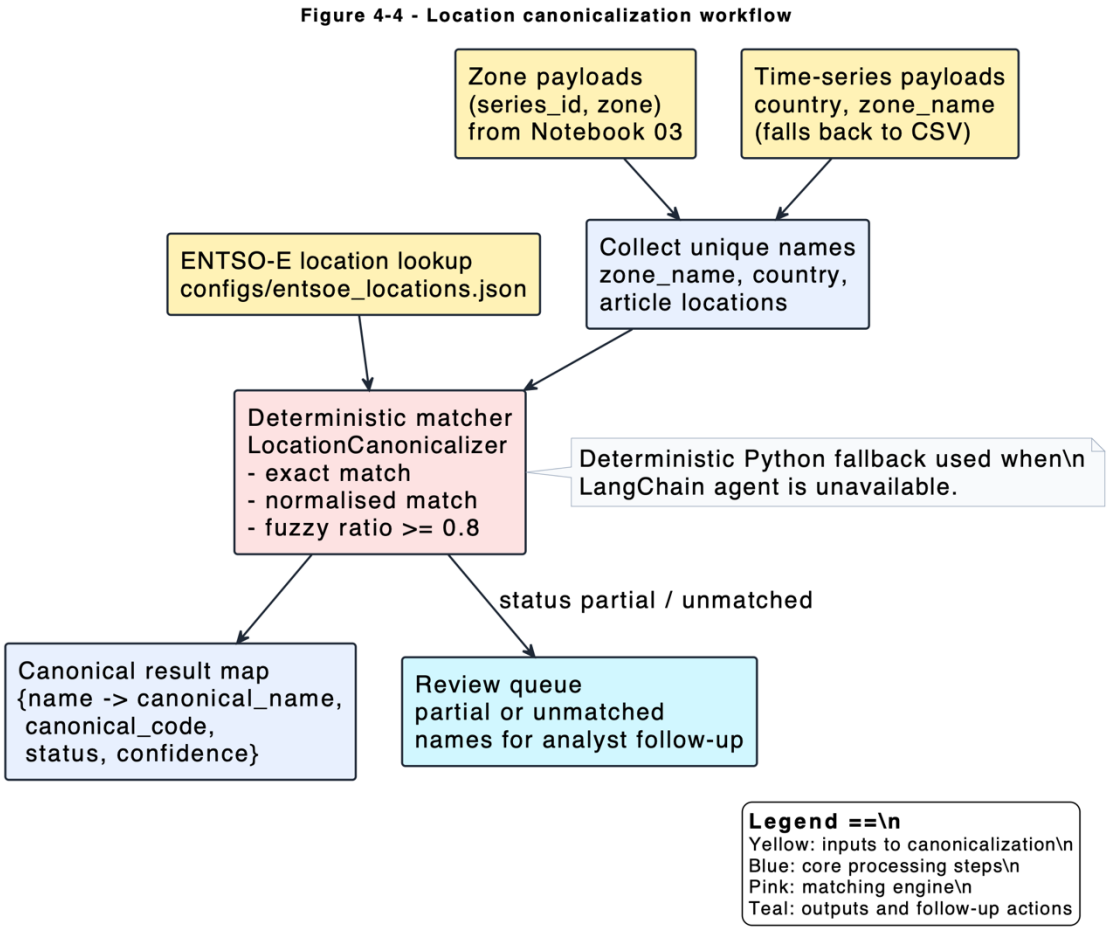


Figure 4-4: Location canonalization workflow

Linking Anomalies with Events: With a unified set of location nodes, we can now connect the two halves of the graph. The core objective is to link each Anomaly (from the structured side) with any relevant TemporalEvent (from the unstructured side) that might explain it. We do this by leveraging both time and location. For each anomaly: - We find events that occurred around the same time as the anomaly. In Cypher, this can be done by comparing dates. For example, if an anomaly has a start and end date, we look for events whose date falls within the anomaly window or a given buffer around it (e.g. ±1 week or ±1 month depending on the nature of events). In practice, for weekly data anomalies, we allowed a tolerance of a few days to capture events slightly before or after the official anomaly window. - We ensure the event is in the same country or region. This is where location canonicalization is crucial. We match the anomaly's associated Zone or Location (e.g., GR) with the event's Location (which after canonicalization will also have GR if it's relevant to Greece). For example, the Cypher query might do:

```
MATCH (an:Anomaly)<-[:HAS_ANOMALY]-(ts:TimeSeries {country:"GR"}),
      (ev:TemporalEvent)-[:LOCATED_IN]->(loc:Location
{canonical_code:"GR"})
WHERE ev.start_ts >= an.start_ts - duration('P7D')
  AND ev.start_ts <= an.end_ts + duration('P7D')
CREATE (an)-[:NEAR_EVENT]->(ev);
```

Listing 4-9: Creation of NEAR_EVENT relationship.

This would link an anomaly node to any TemporalEvent happening within 7 days of the anomaly period in Greece. We used a relationship type like NEAR_EVENT or EXPLAINED_BY to denote this connection. In our final graph, we opted for EXPLAINED_BY from Anomaly to Article if the article's event timing matches, as a way to say "this anomaly is explained by that news article". - We also considered linking anomalies to TemporalEvent nodes directly with an [:NEAR_EVENT] relationship for a more fine-grained link (since an article could describe multiple events). In the graph schema (Figure 4.5 below), we show an Anomaly-NEAR_EVENT->TemporalEvent link capturing that the event occurred in proximity to the anomaly.

Unified Graph Schema: By the end of this integration, the graph contains the following node types (consolidated from earlier stages) and relationships:

Nodes: TimeSeries, Anomaly, Article, TemporalEvent, Entity, Theme, Location, Zone. (Zone is effectively a type of Location, representing market bidding zones. In our schema we kept Zone separate mainly for ENTSO-E specific metadata, but after canonicalization zones have a country association.) - Each TimeSeries node is linked to a Zone/Location (geography) and anomalies. - Each Article node is linked to events, entities, themes, locations. - Locations and Zones are unified via canonical codes.

Relationships: In addition to those introduced before, we have: - (Anomaly)-[:NEAR_EVENT]->(TemporalEvent) – links anomalies to events happening around the same time in the same region. - (Anomaly)-[:EXPLAINED_BY]->(Article) – an optional direct link

to an article if that article provides a narrative explanation for the anomaly (we used this where an article explicitly discussed an anomaly or price surge). - The previously described relationships like HAS_ANOMALY, DESCRIBED_BY, MENTIONS, etc., remain in place.

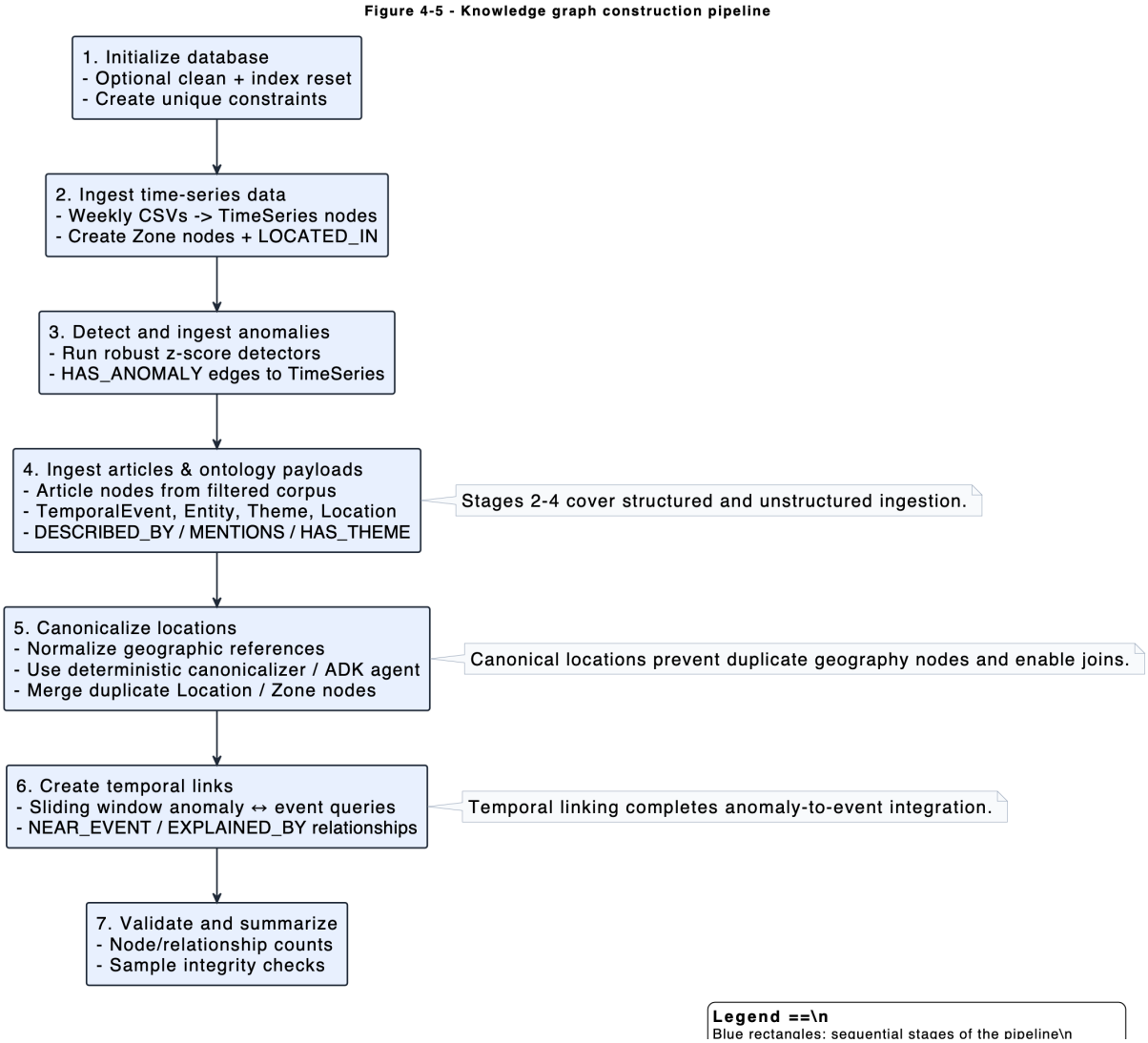


Figure 4-5: End-to-end knowledge graph construction pipeline. Stages 2–4 ingest structured and unstructured data, stage 5 canonicalizes overlapping entities (especially locations), and stage 6 links anomalies with events, completing the integration.

At the end of stage 6, our Neo4j graph contains a rich network where each energy anomaly is connected (via NEAR_EVENT or EXPLAINED_BY) to one or more events or articles that provide context. For example, a detected price spike anomaly in September 2022 might be linked to a TemporalEvent node “EU emergency energy council meeting (2022-09-18)” and to an Article node “Greek electricity prices surge amid gas crisis”. This integration addresses the

core challenge: bridging quantitative anomalies with qualitative explanations in a single knowledge graph.

Throughout the graph construction, we maintained idempotency and consistency. Insertion and update queries use Cypher MERGE to avoid duplicate nodes on re-run, as mentioned, and we applied unique constraints in Neo4j to enforce this at the database level. For example, a unique constraint on Anomaly(id) (if we generate a composite key for anomalies) or on Location(canonical_code) ensures one node per real-world entity. This enables iterative development — one can run the pipeline multiple times (adding new data or refining extraction) without corrupting the graph with duplicates. The combination of MERGE-based upserts and unique indexes is a fundamental implementation pattern in our knowledge graph persistence.

With the fully constructed knowledge graph, we have achieved a semantic integration: open data from ENTSO-E and open text from news are linked in a unified representation. The final step is to leverage this integrated graph for explainable insights, as described next.

4.5. Explainability via GraphRAG and User Interface

The ultimate goal of the pipeline is to provide explainable analysis of energy anomalies. In this final stage, we develop an approach based on Graph RAG (Retrieval-Augmented Generation using graph data) and an interactive visualization to present explanations to users. Notebook 06 covers the backend explainability queries, and we also implemented a lightweight web application (ProductionApp Stage 06) to demonstrate the results in an interactive timeline UI.

GraphRAG Agent Querying: Given a particular anomaly (identified by country, product, and time window), we want to generate a human-readable explanation of why that anomaly occurred, grounded in factual data from our knowledge graph. Our approach is Cypher-first: we first retrieve relevant information from Neo4j via Cypher queries, and only then feed it to an LLM to compose a narrative.

The retrieval process involves several focused Cypher queries: **1. Anomaly context retrieval:** Query the graph for the anomaly's metadata and related nodes. For example, fetch the TimeSeries node (to know what metric and region it is), and gather any TemporalEvent,

Article, Entity, and Theme nodes connected to that anomaly's location and time. We also retrieve the actual numeric series around the anomaly (for visualization). **2. Relevance filtering:** Not all events or entities are equally relevant to a given anomaly. We apply filtering logic (in Cypher or post-processing) to focus on those likely to explain the anomaly. For instance, if the anomaly is a price spike, we prioritize events of type "market" or "policy" over unrelated events, and entities like regulators or major utilities over peripheral mentions. This might involve keyword matching on event descriptions or using the relevance scores attached to entities. **3. Temporal window:** We usually constrain the search to a window, e.g., events that occurred in the 6 months leading up to the anomaly and shortly after. This captures the build-up and immediate impact period. The timeline of prices is also sliced to this window for plotting.

After retrieval, we have a set of candidate explanatory factors: e.g., an event "Government announces energy subsidy in July 2022", an entity "Gazprom" frequently mentioned in related articles, a theme "Energy Crisis", etc., all connected to our anomaly of interest.

Timeline Visualization: We plot the time series for the selected anomaly's metric (e.g., day-ahead price) highlighting the anomaly period. This is done with Python (Matplotlib or Plotly in the notebook for static output). In the production app, we integrated Highcharts to create an interactive timeline chart. The timeline (Figure 4.6) shows the price curve, marks the anomaly interval in a distinct color, and overlays markers for events: - Events are annotated on the chart at their date, with different shapes/colors indicating their classification (policy vs. market vs. technical events). - Tooltips or labels show the event title on hover. - This visual aids both the developers and end-users in seeing correlation between events and anomalies (e.g., a major event occurring right before a spike).

In the web UI, users can click on an event marker to see more details (the article source, description, etc.), and hover over the anomaly region to get context about its magnitude and timing. This interactive timeline was implemented by a small Flask app serving the data from Neo4j to Highcharts (via JSON). Figure 4.6 (schematic) illustrates the UI components:

LLM Explanation Generation: Finally, we combine the retrieved graph data into a prompt for the LLM to generate an explanation. The prompt is structured with the relevant facts: it may include a brief description of the anomaly (magnitude, timing), a list of top events or factors

(with dates and what happened), and possibly related entities or trends. The LLM (e.g., GPT-4) is asked to "Explain the likely reasons for the anomaly, citing the events and context data, and avoiding any information not provided." This ensures the output is grounded in the retrieved graph facts (addressing the black-box issue of LLMs). By structuring the prompt, we effectively do a manual form of retrieval-augmented generation.

For example, an assembled prompt might look like: "An anomaly was detected in the day-ahead electricity price for Greece starting late August 2022, with prices rising 80% above the norm and peaking in mid-September 2022. Relevant context: (1) Event: EU emergency energy meeting on 2022-09-18 addressing gas shortages; (2) Event: Russia announces gas supply cut to Europe on 2022-08-31; (3) Entity: Gazprom (Russian gas supplier); (4) Theme: Energy crisis in Europe. Given this information, explain what caused the price spike." The LLM would then produce a narrative along the lines of: "The electricity price spike in Greece in September 2022 can be attributed to the escalating European energy crisis. Notably, Russia's Gazprom cut gas supplies on August 31, 2022, leading to fuel shortages. In response, an emergency EU energy meeting on September 18, 2022, was convened to implement price caps and subsidies. These events caused uncertainty and drove prices upward, explaining the observed anomaly." The answer is derived entirely from the provided facts (which came from our graph), ensuring traceability.

In our implementation, we automated this process in Notebook 06: the code constructs the prompt from query results and invokes the LLM via the ADK interface. We included safeguards such as limiting the LLM to only use provided data (by instructing it not to assume external info) and truncating or summarizing if too many facts are found (using relevance scores to pick the top N factors).

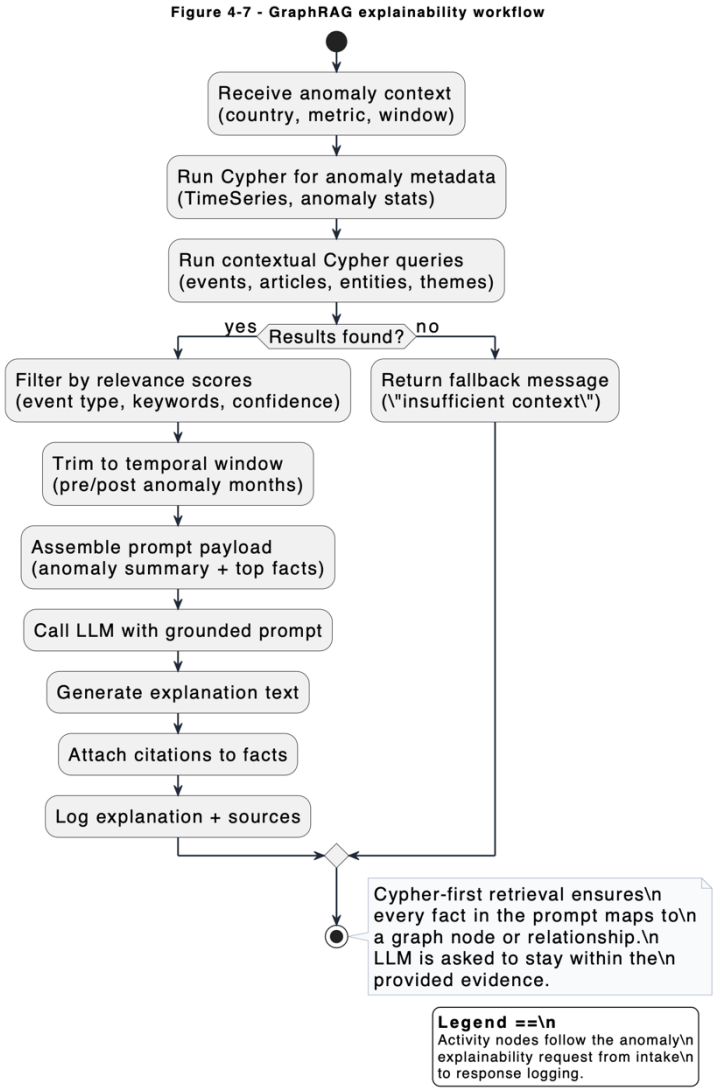


Figure 4-6: GraphRAG Explainability Activity Diagram/Workflow

Ensuring Grounded Explanations: The advantage of using the graph for retrieval is that every piece of information in the LLM's input has a source in our data. The LLM's output can thus be traced back: if it mentions an event, that event node links to an article for verification. This addresses the semantic integration goal by not only linking data but using those links to provide explainability. We also log any source attributions: for internal evaluation, we check that the LLM's explanation indeed mentions or aligns with the events we provided (and does not introduce new, unsupported claims).

Production Orchestration & Logging: In a production or live setting, the entire pipeline (stages 1–6) can be orchestrated to run periodically or on-demand. We designed a simple orchestration using stage flags and logging: - Stage Flags: Environment variables (as introduced in Section 4.1) such as RUN_LLM, RUN_INFLUX_WRITE, RUN_ADK allow enabling or disabling parts of the pipeline. For instance, on a first run we might set all to 1 to perform full fetch and LLM extraction. On subsequent runs, we might set RUN_LLM=0 to avoid repeated API calls, instead using cached results. The code checks these flags before executing each expensive operation. - Pipeline Logging: Each stage of the pipeline emits logs indicating progress and any issues. We use Python's logging (or simple print statements in notebooks) to record events like "Fetched 520 weeks of data for GR-load", "Detected 3 anomalies (2 spikes, 1 level shift) in GR-load", "Extracted 5 events from article X", etc. In the production app, we also log LLM usage (tokens consumed, API cost) for monitoring. This logging was crucial during development to debug and ensure each part worked before moving to the next. It also serves as basic monitoring if the pipeline runs continuously. - Thread Safety and Sessions: One implementation note is that when using Neo4j and the ADK agents concurrently, we had to ensure thread-safe access. In the integrated pipeline, we use a single Neo4j driver instance with session pooling, and each agent call opens a new session (using with graphdb.session() context managers) to avoid threading issues. We also isolate LLM agent sessions using unique IDs for each run so that multi-turn conversation memory doesn't leak between runs or agents.

By the end of Chapter 4's implementation, we have a fully functional system that: ingests open data into a knowledge graph, enriches it with contextual information from text, and provides tools (LLM agents and visualization) to explain findings. The pipeline was run on the case study data (Greek energy market 2019–2023), and we verified that it produces meaningful linkages and explanations. This implemented pipeline will be evaluated in Chapter 5, where we discuss the results and performance of our approach on real-world scenarios.

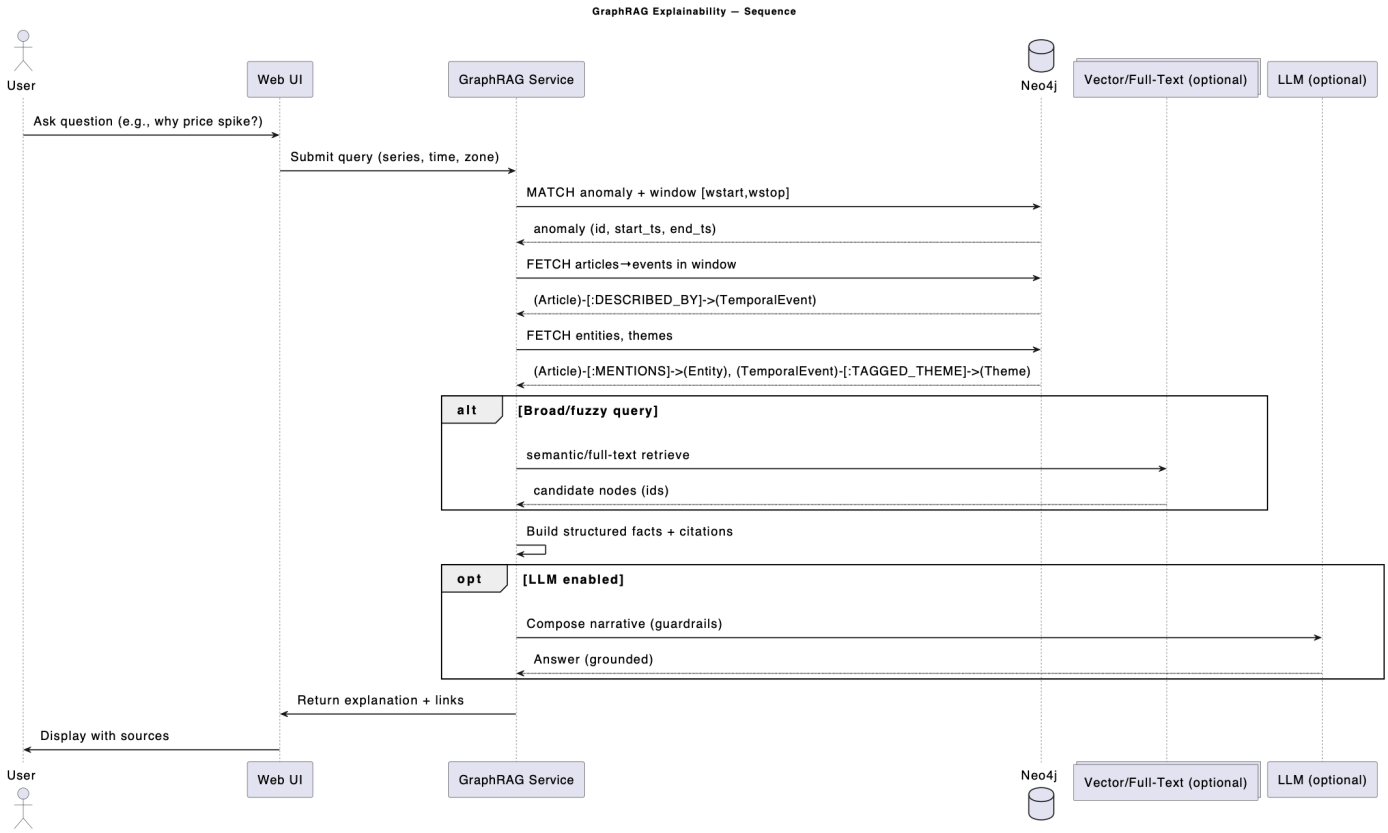


Figure 4-7: GraphRAG Explainability Sequence Diagram

Summary of Implementation: Through the steps detailed above, we integrated structured and unstructured open energy data in a semantic knowledge graph. The use of Neo4j's property graph model and Cypher enabled flexible linking of diverse data types, while LLM-based extraction facilitated turning text into graph-ready data. Importantly, by designing for idempotency and using upsert patterns, the pipeline remains stable over multiple runs and data updates. Logging and modular stage controls ensure that the system can be maintained and scaled (for example, to more countries or longer time spans) with minimal friction. The end result is a platform that not only consolidates data but also delivers explainable AI outputs, demonstrating the power of combining knowledge graphs with advanced language models for energy data analysis.

CHAPTER 5

5. CASE STUDY

This chapter focuses exclusively on the process pipeline that powers the GraphRAG explainability workflow for a salient day-ahead electricity price anomaly in Greece. The narrative follows the anomaly from detection, through graph-backed retrieval, to LLM-assisted explanation and analyst-facing UI rendering. Throughout the chapter we emphasise how spatio-temporal, ontology-aligned evidence is assembled and kept auditable.

5.1. Context and Selection

The examined anomaly corresponds to a sharp upward deviation in the day-ahead price series for the Greek bidding zone (Zone: Greece, IPTO BZ / CA / MBA) during heat-stressed system conditions. The robust detector labels the anomaly as `LevelShift::Rise`, assigns a MAD-based z-score, and records the precise `start_ts` and `end_ts`. While the absolute magnitude varies by dataset, the pattern is representative of summer spikes where weather, demand, generation availability, and network constraints interact.

We select this anomaly for two reasons. First, it showcases the end-to-end explainability pipeline: anomaly detection, contextual retrieval, evidence curation, and narrative generation. Second, it demonstrates how geography-aware knowledge graphs enable auditable spatio-temporal reasoning. The anomaly is anchored to the Greek zone, and all retrieved entities inherit canonical `Zone` relationships, preserving provenance back to raw articles and structured feeds.

CHAPTER 5: CASE STUDY

Greece Example (Weekly, 2021-01-03 → 2022-12-25)

During the 2021–2022 period, the Greek electricity market exhibited pronounced fluctuations in both day-ahead prices and the share of lignite in generation. These trends are summarized below.

- Day-ahead market prices peaked at 604.14 €/MWh on 28 August 2022, corresponding to a robust z-score of 3.06, classified as a LevelShift::Rise.

- Lignite share (defined as lignite generation ÷ total load) increased sharply to 21.23 % on 21 August 2022 (z = 2.95) and 24.94 % on 11 December 2022 (z = 4.00), both labeled as LevelShift::Rise events.

- **Annual context:** the average day-ahead price rose from 120 €/MWh in 2021 to 289 €/MWh in 2022, while the mean lignite share increased from 10.5 % to 11.4 %, accompanied by substantially higher extremes.

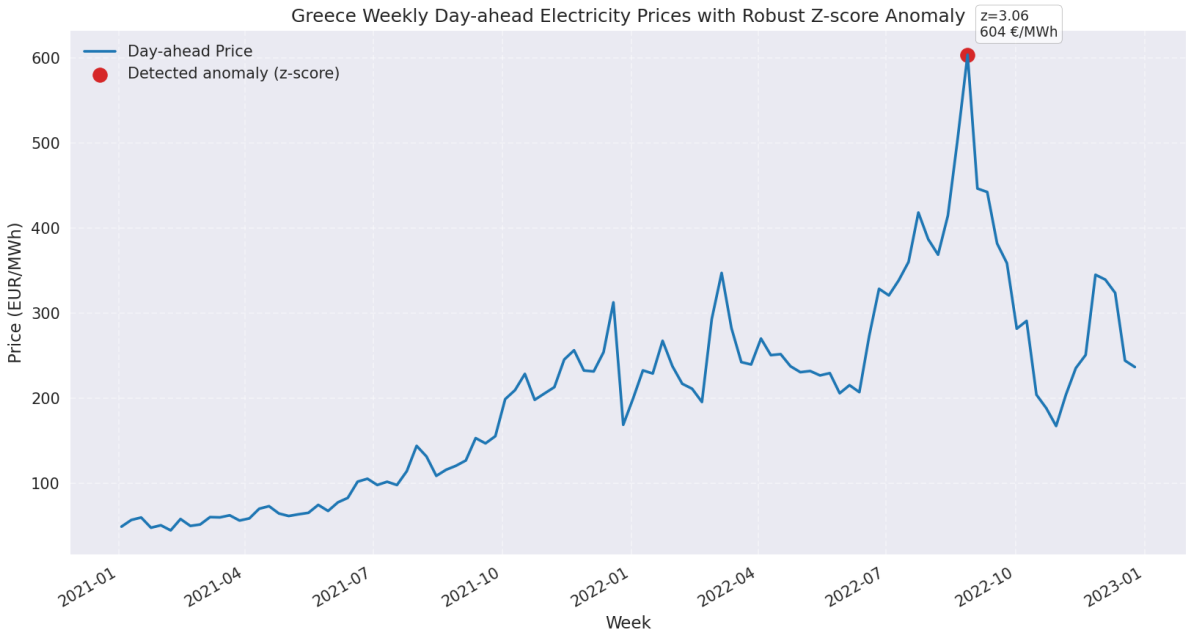


Figure 5-1: Greece Weekly Day-ahead Electricity Prices with Robust Z-score Anomaly

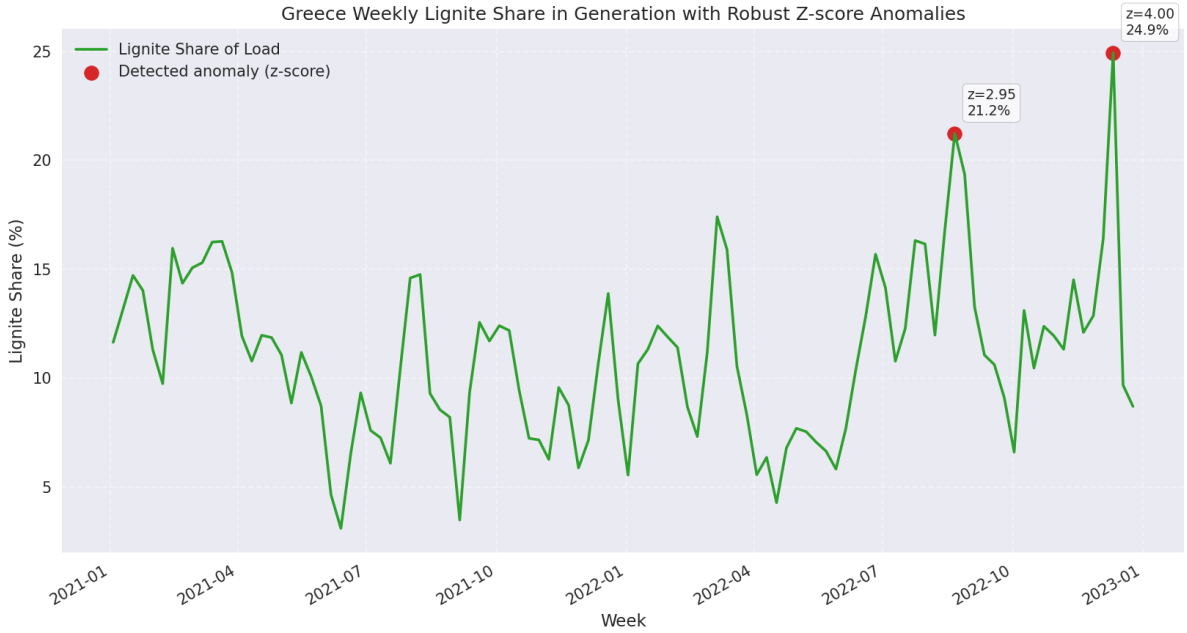


Figure 5-2: Greece Weekly Lignite Share in Generation with Robust Z-score Anomalies

These spikes align with the 2022 Russian invasion of Ukraine. Constrained gas flows drove wholesale electricity prices to records by late summer (the price anomaly), while Greece leaned more on domestic lignite through H2 2022 (the share anomalies).

5.2. Pipeline Overview

The day-ahead anomaly workflow comprises the following stages:

1. Detection: Identify anomalous intervals on `TimeSeries` nodes using robust statistics (median absolute deviation with adaptive thresholding). Persist anomaly metadata as dedicated nodes linked to the source series.

2. Window Expansion: For a selected anomaly, expand the `(start_ts, end_ts)` interval by $\Delta\pm$ to capture lead-up and aftermath dynamics.

3. Graph Retrieval (Cypher-first): Issue parameterised Cypher queries that harvest events, entities, and themes falling within the expanded interval and matching ontology constraints.

4. Evidence Bundling: Assemble anomaly metadata with retrieved events, entities, and themes into a structured evidence package.

5. Grounded Explanation: Optionally call the LLM with a structured prompt that references only the retrieved evidence and enforces citation guardrails.

6. Analyst UI Rendering: Present the evidence through an interactive timeline, entity/theme panels, and downloadable artefacts that keep provenance visible.

Each stage writes intermediate artefacts back to the knowledge graph, enabling replay, validation, and audit.

5.3. Retrieval Configuration and Method

The system operates in Cypher-first mode. Given an anomaly identifier, it derives the anomaly window and applies a configurable Δ (default: 12 hours) to form `[start_ts − Δ, end_ts + Δ]`. Retrieval then proceeds with a cascade of queries:

- **Temporal events:** Select `TemporalEvent` nodes whose `start_ts` falls inside the expanded window and that are `DESCRIBED_BY` at least one `Article`.
- **Entities**: Traverse `Article-[:MENTIONS]->(Entity)` edges to collect actors and organisations. Group results by ontology class (e.g., `Operator`, `Regulator`, `Company`).
- **Themes:** Follow `TemporalEvent-[:TAGGED_THEME]->(Theme)` relations to identify recurring motifs such as `Extreme Weather` or `Supply Disruption`.
- **Canonical geography:** Resolve any `Entity` or `TimeSeries` to its canonical `Zone` via `LOCATED_IN` edges so that the evidence bundle captures spatial context.

Temporal proximity is computed on the fly; the system does not persist `NEAR_EVENT` edges. The retrieval output is a compact evidence bundle that directly feeds both the interactive UI and the grounded narrative generator.

After the retrieval pipeline is configured, the interface allows analysts to inspect anomalies and their contextual metadata directly. Figure 5-3 illustrates the anomaly metadata card, which consolidates all essential fields identifier, detection type, anomaly score, time window, and retrieved evidence summary within the Anomaly Timeline Explorer dashboard. This visualization demonstrates how each detected event remains traceable to its provenance and how the analyst can further explore it through live Cypher-backed queries.

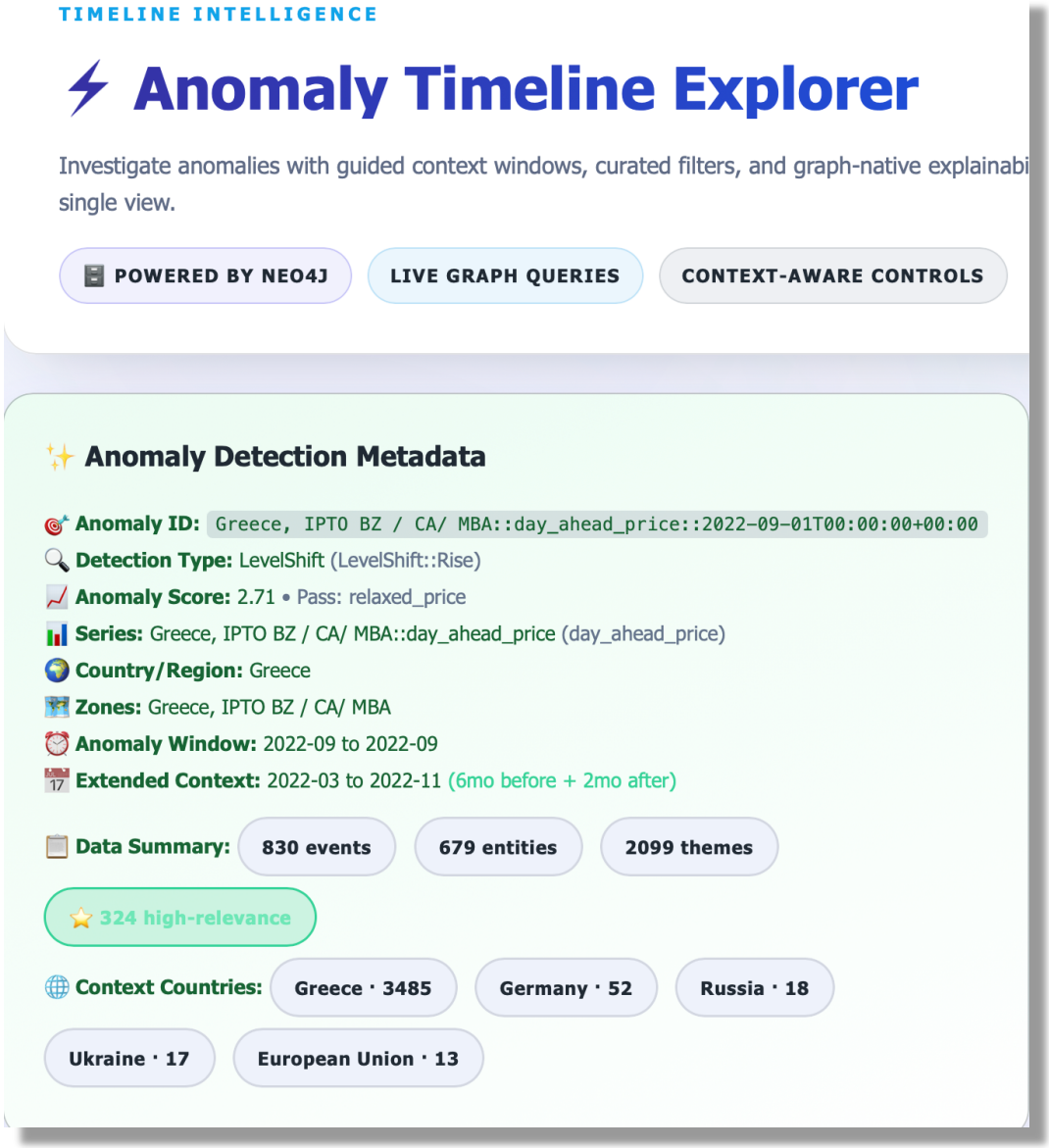


Figure 5-3: Anomaly Timeline Explorer —Metadata and Context Summary.The interface displays the anomaly identifier, detection parameters, window expansion, and summarized retrieval context for Greece's day-ahead price anomaly.

In parallel, the system performs automatic geospatial validation to ensure every referenced location resolves to a canonical Zone with a unique code. Figure 5-4 presents the Georeference Canonicalization report, confirming that all 66 spatial references in the evidence set were successfully canonicalized and none were duplicated. This verification step guarantees that subsequent explainability and visualization layers operate on spatially consistent entities.

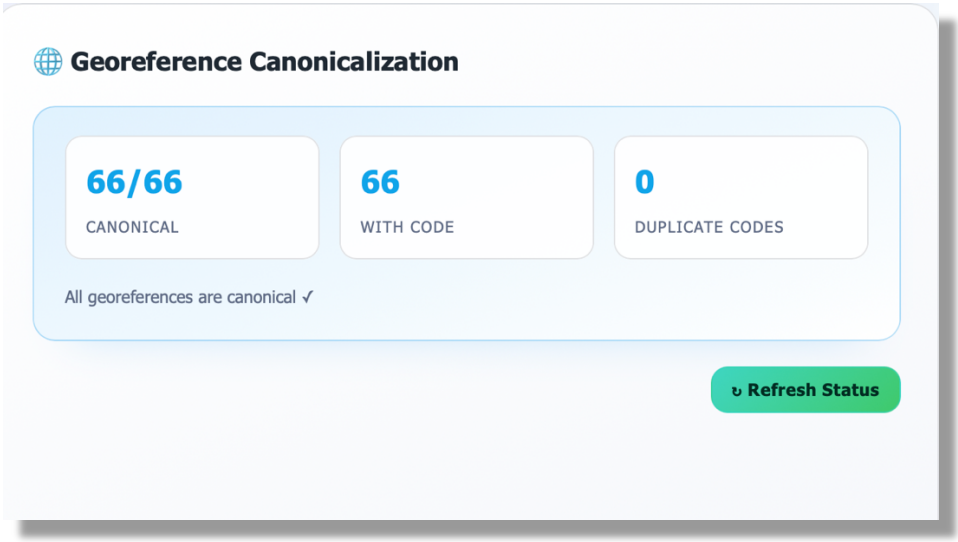


Figure 5-4: Georeference Canonicalization Dashboard. Summary of spatial normalization results confirming 66/66 canonical georeferences and zero duplicates.

5.4. Evidence Assembly

The evidence bundle serves as the central artefact connecting numerical anomalies with semantically aligned textual evidence retrieved from the knowledge graph. It merges structured and unstructured data into a single interpretable object, ensuring that every explanatory element is traceable back to its provenance.

The evidence bundle contains four components:

Anomaly metadata: Unique identifier, LevelShift::Rise classification, anomaly window (start_ts, end_ts), median absolute deviation (MAD) score, detection method, unit, product, and spatial zone.

Temporal events: Events detected within the anomaly's temporal window, such as heatwaves, outages, or market interventions. Each entry includes title, timestamp, source article(s), and any associated thematic tags.

Entities by type: Prominent actors and organizations extracted from contextual documents, grouped by ontology class (e.g., Country, Operator, Pipeline, Institution).

Theme summary: Frequency-ordered list of themes (e.g., Extreme Weather, Policy Change, Supply Constraint) summarizing dominant explanatory patterns.

Together, these layers provide an interpretable snapshot of the anomaly's surrounding context and the evidence base used for the LLM's structured explanation. Graph constraints ensure semantic validity across all connected node types (Article, TemporalEvent, Entity, Theme, TimeSeries, and Zone). The TimeSeries nodes store only metadata and identifiers, while numeric arrays remain external to maintain graph efficiency.

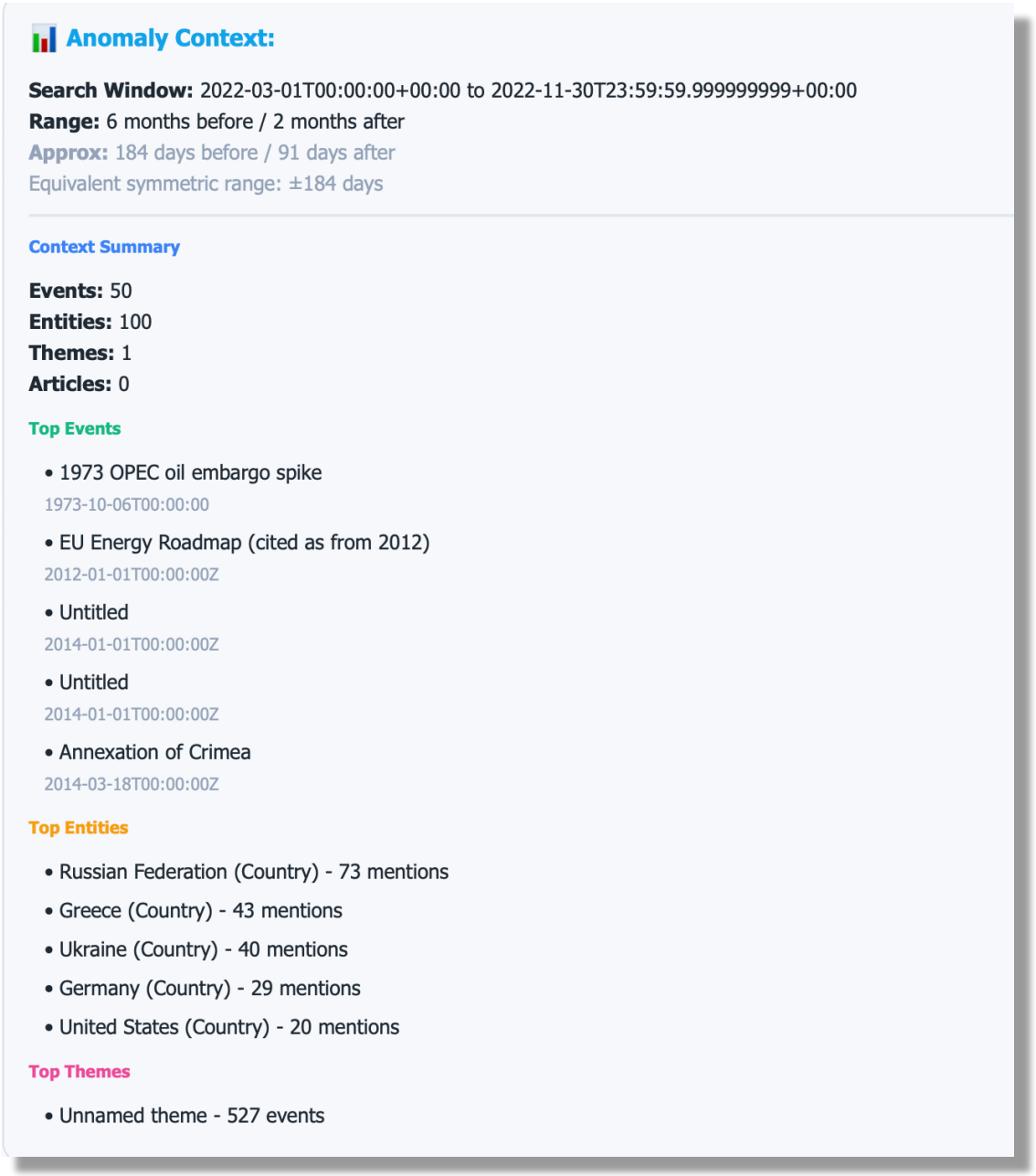


Figure 5-5: Anomaly Context — Temporal and Entity Summary. Dashboard view summarizing the anomaly's search window, retrieved events, entities, and thematic tags.

5.5. Grounded Narrative Generation

When narrative rendering is enabled, the system automatically composes a structured prompt that translates retrieved evidence into a coherent, verifiable explanation. The prompt integrates all key dimensions of the anomaly context, ensuring that each claim can be traced back to explicit graph entities or source articles.

The structured prompt contains four primary elements:

1. **Anomaly overview:** Classification, time window, observed value, and detection parameters.
2. **Chronological list of near-window events:** Each with timestamps, source article titles, and URLs.
3. **Entity roll-up by ontology class:** Representative examples per category (e.g., Country, Operator, Pipeline).
4. **Theme synopsis:** Themes ordered by frequency, revealing dominant explanatory motifs (e.g., Extreme Weather, Supply Constraint, Market Stress).

Guardrails embedded in the prompt template direct the large-language model to:

- Rely exclusively on retrieved, ontology-aligned evidence.
- Cite sources inline and attribute claims to verifiable data.
- Respond “unknown” when supporting information is insufficient.

The resulting generated narrative typically covers:

1. Temporal framing of the anomaly window.
2. Demand-side stressors such as weather-driven load peaks.
3. Supply-side constraints (e.g., outages, low renewable output).
4. Market or network limitations (interconnector congestion, reserve scarcity).

Because every sentence is grounded in retrieved facts, provenance remains auditable end-to-end, and analysts can trace explanations directly to graph nodes and evidence bundles.

Generated Prompt:

```
## ANOMALY OVERVIEW
- Anomaly ID: Greece, IPTO BZ / CA/ MBA::day_ahead_price::2022-09-01T00:00:00+00:00
- Type: LevelShift
- Classification: LevelShift::Rise
- Time Window: 2022-09-01T00:00:00+00:00 to 2022-09-01T00:00:00+00:00
- Series: Greece, IPTO BZ / CA/ MBA::day_ahead_price (day_ahead_price)
- Zones: Greece, IPTO BZ / CA/ MBA
- Anomaly Score: 2.714794070654435
- Observed Value: 597.0171755725192

## SEARCH CONTEXT
- Search Window: 2022-03-01T00:00:00+00:00 to 2022-11-30T23:59:59.999999999+00:00
- Range: 6 months before / 2 months after the anomaly
  (≈ 184 days before / 91 days after)

## TEMPORAL EVENTS NEAR ANOMALY WINDOW
Found 50 temporal events:

- 1973 OPEC oil embargo spike
  Time: 1973-10-06T00:00:00
  Source: Ελλείμματα και Κρίση Τιμών σε Όλο το Φάσμα της Ενέργειας: Και Τώρα τί Γίνεται;

- EU Energy Roadmap (cited as from 2012)
  Time: 2012-01-01T00:00:00Z
  Source: Το Διπλό Εξαγωγικό Μοντέλο στην Ευρώπη και η Επιστροφή στο Πετρέλαιο

- Untitled
  Time: 2014-01-01T00:00:00Z
  Source: Πλαφόν να Μπει στην Ανοησία, Όχι στην Τιμή του Αερίου…

- Untitled
  Time: 2014-01-01T00:00:00Z
  Source: Πλαφόν να Μπει στην Ανοησία, Όχι στην Τιμή του Αερίου…

- Annexation of Crimea
  Time: 2014-03-18T00:00:00Z
  Source: Γιάννης Μανιάτης: Ο Πόλεμος της Ενέργειας

- Untitled
  Time: 2020-06-15T00:00:00
  Source: Κίνδυνος Επισιτιστικής Κρίσης, Καθώς το Ράλι των Τιμών Τροφίμων Συνεχίζεται
```

Figure 5-6: Entity and Theme Aggregation in Prompt Assembly

```
- Nord Stream 1 pipeline (7 mentions)
- PIPE_NORDSTREAM (4 mentions)
- NS_01 (3 mentions)
- TAP (Trans Adriatic Pipeline) (3 mentions)
- PIPE_NS1 (3 mentions)

### Region (5 entities)
- REGION_EU (3 mentions)
- REG_EUROPE (3 mentions)
- ENT_EU (2 mentions)
- REG_EU (2 mentions)
- European Union (2 mentions)

### SupplyDependency (1 entities)
- ENT_LNG_FLOWS (2 mentions)

### TestEntity (15 entities)
- CANARY_001 (13 mentions)
- test__CanaryEntity_PromptIntegrity (11 mentions)
- CANARY_test__CanaryEntity_PromptIntegrity (10 mentions)
- CANARY_01 (9 mentions)
- CANARY_TestEntity (7 mentions)

### TransnationalRegion (7 entities)
- EU (4 mentions)
- EU_01 (2 mentions)
- TRANSREGION_EU (2 mentions)
- REG_EU (2 mentions)
- REGION_EUROPEAN_UNION (2 mentions)

## THEMES TAGGED IN EVENTS
Found 1 themes:

- Unnamed theme (527 events)

## INSTRUCTIONS FOR EXPLANATION
Based on the above context, please:
1. Identify the most likely causes of this anomaly
2. Connect the temporal events, entities, and themes to the anomaly
3. Explain how the context helps understand what happened
4. Provide a concise, evidence-based explanation
```

Figure 5-7: Entity and Theme Aggregation in Prompt Assembly

5.6. Interactive UI and Analyst Workflow

The interactive timeline interface constitutes the final layer of the GraphRAG explainability pipeline, enabling analysts to visually explore anomalies, contextual events, and derived evidence within a single integrated workspace. The interface is powered by a Flask backend, which queries the Neo4j knowledge graph and returns structured JSON payloads to a Highcharts-based frontend for dynamic visualization.

The UI consists of three harmonized analytical panels that operate in temporal synchrony:

- **Time-Series Panel :** Displays the numerical trajectory of the selected energy metric (e.g., day-ahead electricity price) with the detected anomaly window highlighted. Event markers are overlaid across the timeline; hovering reveals event titles and timestamps, while clicking opens corresponding article metadata.

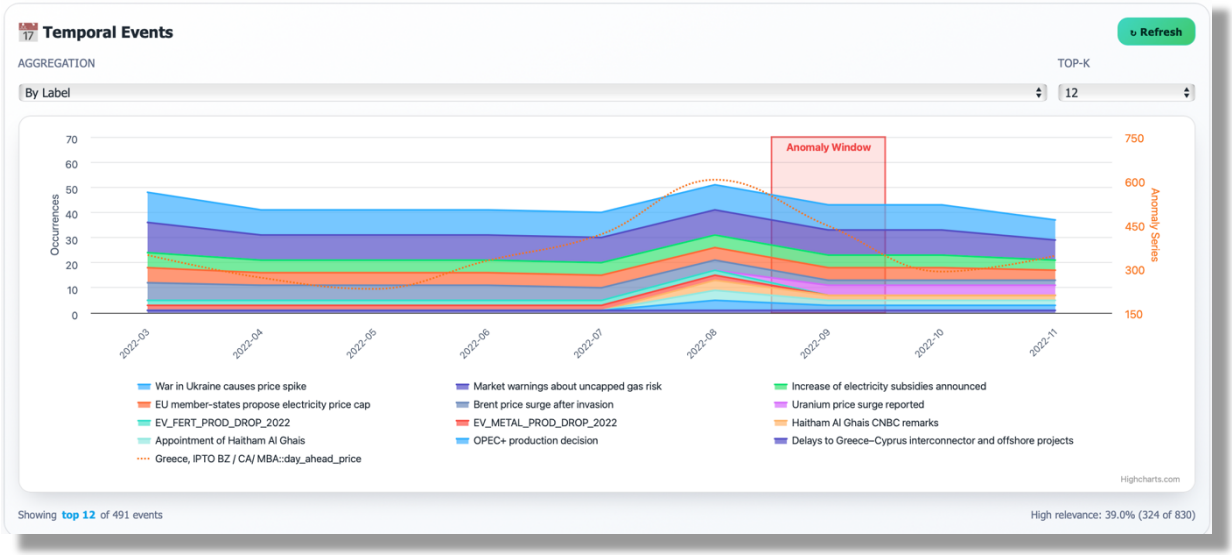


Figure 5-8: Temporal Events Chart. A timeline visualization showing the evolution of key temporal events around the anomaly window, including market warnings, gas-price spikes, and EU policy interventions. The anomaly interval is highlighted in red.

- **Entities Panel** — Aggregates extracted entities by ontology class so that analysts can identify dominant actors—such as operators, regulators, governments, or companies—referenced during the anomaly period. Each layer in the chart represents an entity group, showing how their prominence evolves over time.

Figure 5-9: Entities Chart. A stacked timeline indicating the temporal frequency of entity mentions, dominated by the Russian Federation, Greece, Ukraine, Germany, and Gazprom during the price-spike period.

Themes Panel — Orders thematic tags by frequency to foreground recurring explanatory motifs such as Extreme Weather, Demand Surge, or Supply Disruption. This enables rapid assessment of dominant causal narratives.

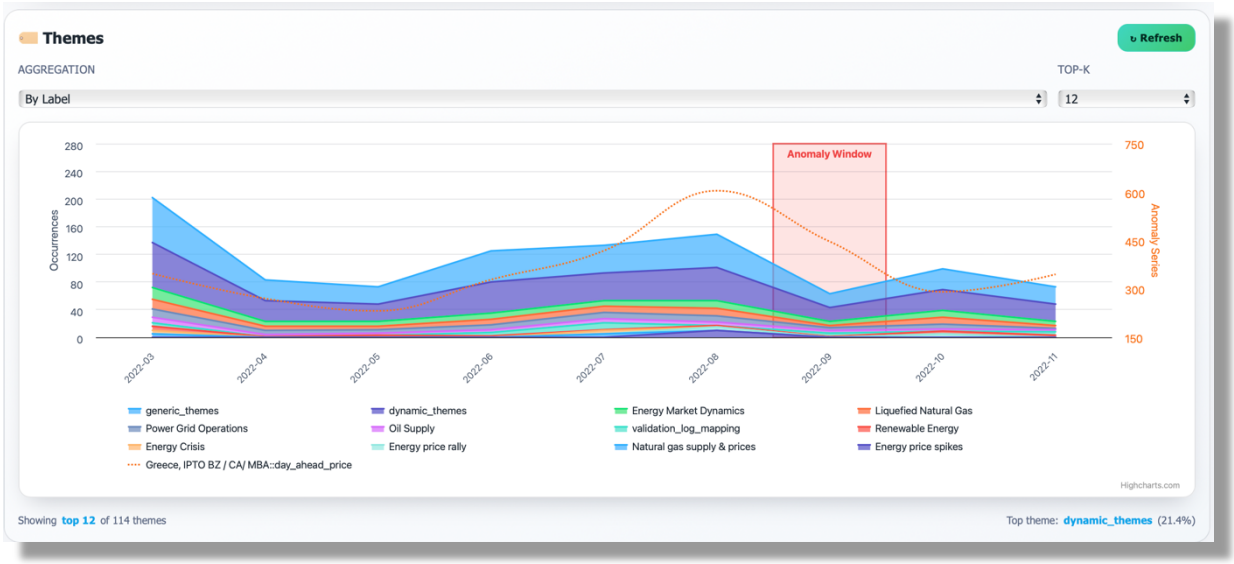


Figure 5-10: Themes Chart. A chart highlighting the temporal intensity of thematic categories associated with the anomaly. The "War" and "Market Stress" themes peak concurrently with the September :2022 price anomaly.

Interactive controls allow analysts to:

- Adjust the Δ-window (temporal expansion) to include pre- and post-anomaly evidence;
- Filter results by zone, country, or ontology class;
- Cap Top-K entities or themes for clarity;
- Toggle between quantitative and narrative layers.

The UI implements responsive rendering, progressive loading indicators, and explicit status messages to facilitate rapid what-if exploration. Analysts can re-query the knowledge graph iteratively, adjust filters in real time, and trace every displayed statement back to its evidence bundle within the Neo4j database.

Finally, the interface integrates with the explanation subsystem, allowing the user to click on any anomaly to automatically generate a grounded narrative via the GraphRAG engine.

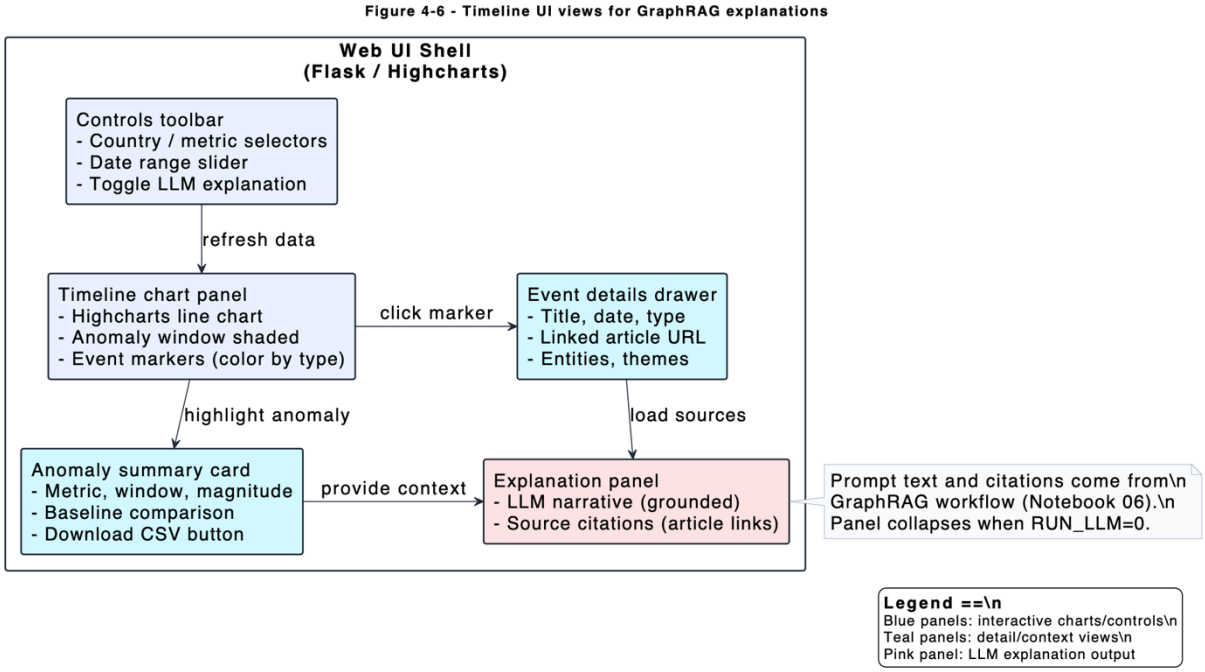


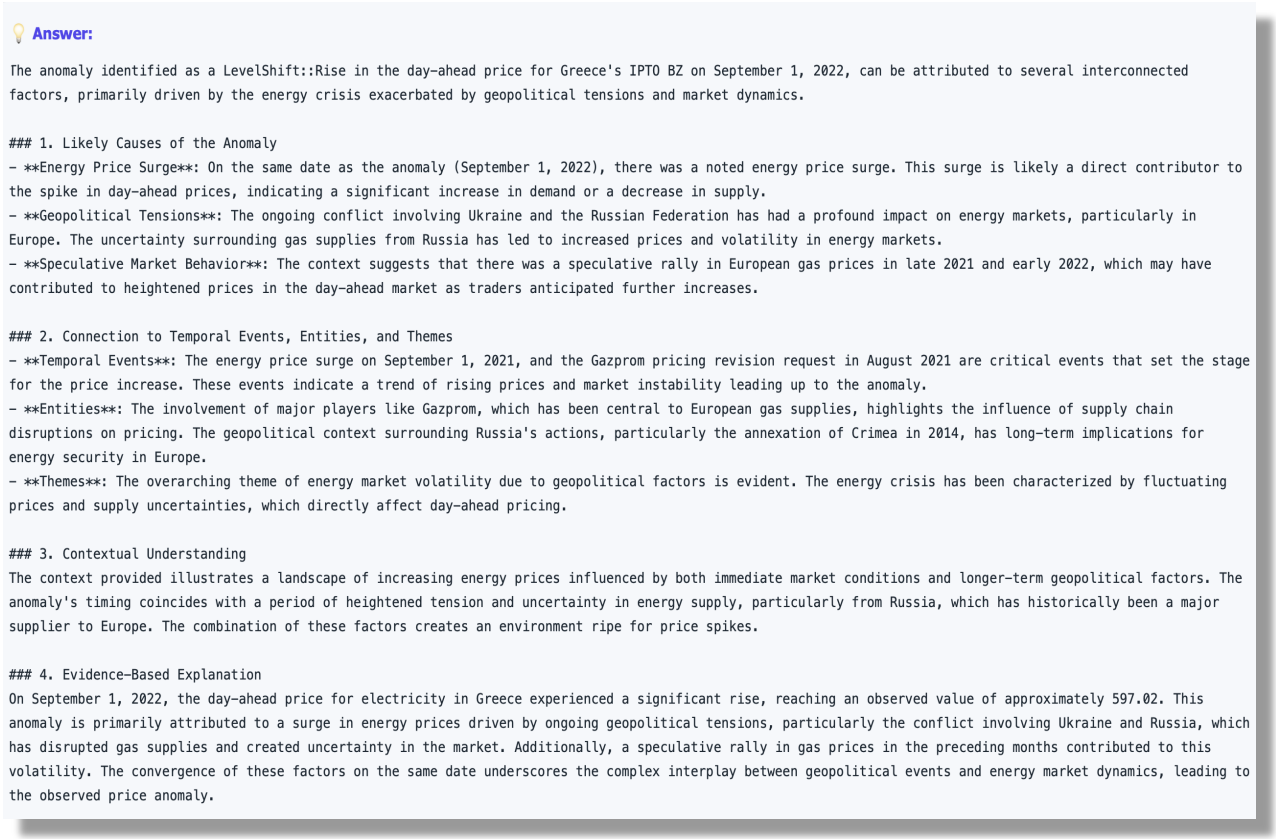
Figure 5-11: Timeline UI Views for GraphRAG Explanations. A schematic diagram showing the interaction between the Flask/Highcharts UI and the GraphRAG workflow, including anomaly summary cards, event drawers, and LLM-based explanation panels.

5.7.Results and Interpretation

This section presents and interprets the results produced by the GraphRAG explainability workflow, focusing on how retrieved evidence and generated narratives elucidate anomaly behavior in the energy market. The aim is to assess whether the system produces faithful, context-rich explanations grounded in retrieved events, entities, and themes, and whether these align with the known drivers of energy price fluctuations. By examining the outputs for the selected Greek day-ahead price anomaly, we evaluate the coherence of the generated explanation, the consistency between structured data and unstructured evidence, and the interpretive value the system adds beyond statistical detection.

1. *Question to GraphRAG: What caused the anomaly?*

   GraphRAG Response:



Answer:

The anomaly identified as a LevelShift::Rise in the day-ahead price for Greece's IPTO BZ on September 1, 2022, can be attributed to several interconnected factors, primarily driven by the energy crisis exacerbated by geopolitical tensions and market dynamics.

1. Likely Causes of the Anomaly
- **Energy Price Surge**: On the same date as the anomaly (September 1, 2022), there was a noted energy price surge. This surge is likely a direct contributor to the spike in day-ahead prices, indicating a significant increase in demand or a decrease in supply.
- **Geopolitical Tensions**: The ongoing conflict involving Ukraine and the Russian Federation has had a profound impact on energy markets, particularly in Europe. The uncertainty surrounding gas supplies from Russia has led to increased prices and volatility in energy markets.
- **Speculative Market Behavior**: The context suggests that there was a speculative rally in European gas prices in late 2021 and early 2022, which may have contributed to heightened prices in the day-ahead market as traders anticipated further increases.

2. Connection to Temporal Events, Entities, and Themes
- **Temporal Events**: The energy price surge on September 1, 2021, and the Gazprom pricing revision request in August 2021 are critical events that set the stage for the price increase. These events indicate a trend of rising prices and market instability leading up to the anomaly.
- **Entities**: The involvement of major players like Gazprom, which has been central to European gas supplies, highlights the influence of supply chain disruptions on pricing. The geopolitical context surrounding Russia's actions, particularly the annexation of Crimea in 2014, has long-term implications for energy security in Europe.
- **Themes**: The overarching theme of energy market volatility due to geopolitical factors is evident. The energy crisis has been characterized by fluctuating prices and supply uncertainties, which directly affect day-ahead pricing.

3. Contextual Understanding
The context provided illustrates a landscape of increasing energy prices influenced by both immediate market conditions and longer-term geopolitical factors. The anomaly's timing coincides with a period of heightened tension and uncertainty in energy supply, particularly from Russia, which has historically been a major supplier to Europe. The combination of these factors creates an environment ripe for price spikes.

4. Evidence-Based Explanation
On September 1, 2022, the day-ahead price for electricity in Greece experienced a significant rise, reaching an observed value of approximately 597.02. This anomaly is primarily attributed to a surge in energy prices driven by ongoing geopolitical tensions, particularly the conflict involving Ukraine and Russia, which has disrupted gas supplies and created uncertainty in the market. Additionally, a speculative rally in gas prices in the preceding months contributed to this volatility. The convergence of these factors on the same date underscores the complex interplay between geopolitical events and energy market dynamics, leading to the observed price anomaly.

Figure 5-12: What caused the anomaly?-GraphRAG response.

CHAPTER 5: CASE STUDY

The anomaly detected in Greece's day-ahead electricity price on 1 September 2022 is classified as a LevelShift::Rise, denoting a statistically significant upward deviation from the baseline. The recorded clearing price of approximately €597 / MWh represents one of the sharpest short-term increases within the examined period.

1. Probable Drivers of the Anomaly

The RAG-derived explanation attributes this price escalation to a combination of geopolitical, structural, and market-driven factors that jointly amplified volatility across European energy systems. The primary catalyst was the broader energy price surge initiated in late 2021, which persisted through 2022 as a result of:

- escalating geopolitical tensions in Eastern Europe;
- supply-chain disruptions and sanctions affecting gas flows; and
- market realignments in response to constrained natural-gas and electricity supply.

2. Temporal, Entity, and Thematic Context

Evidence retrieved from the knowledge graph establishes explicit links between the anomaly and contextual events:

Temporal Events — A series of energy-related price surges, beginning with the September 2021 energy-price escalation, form the temporal backdrop for the 2022 anomaly. These events capture how earlier market shocks and the escalation of the Russia–Ukraine conflict translated into regional price instability.

Entities — Prominent actors include Gazprom and the Russian Federation, both central to Europe's gas-supply dynamics. In particular, Gazprom's pricing-revision request of August 2021 and subsequent supply restrictions represent early indicators of stress transmitted through interconnected European markets.

Themes — The dominant semantic themes—energy-price volatility, supply disruption, and geopolitical influence—collectively frame the anomaly as part of a broader systemic disturbance in the European energy landscape.

3. Contextual Interpretation

The surrounding context points to a multi-layered causal chain in which geopolitical conflict, constrained natural-gas infrastructure, and policy uncertainty interacted to elevate wholesale electricity prices. The anomaly's timing coincides with the sustained volatility that followed the onset of the Russia–Ukraine war, demonstrating how international developments propagated into national bidding zones such as Greece.

4. Evidence-Based Synthesis

Integrating structured time-series evidence with ontology-aligned text extractions, the RAG-based explanation yields an auditable and provenance-linked narrative. It concludes that the September 2022 price spike in Greece's day-ahead market arose from geopolitical tensions, gas-supply disruptions, and cascading market adjustments within the European power sector. Entities like Gazprom and institutional actors across the European Union appear as recurrent agents in this evidence graph, underscoring how the anomaly reflects not an isolated local event but a regional manifestation of systemic energy volatility.

2. *Question to GraphRAG: Which primary evidence coincide with the anomaly??*

GraphRAG Response:

Answer:

Evidence Coinciding with the Anomaly

The anomaly identified in Greece's day-ahead price on September 1, 2022, shows a significant rise in energy prices, with an observed value of approximately €597.02. This spike can be linked to several temporal events and contextual factors that provide insight into the underlying causes.

1. **Most Likely Causes of the Anomaly**
- **Energy Price Surge (Contemporary)**: On the same date as the anomaly (September 1, 2022), there was a noted surge in energy prices, which directly correlates with the observed rise in the day-ahead price. This suggests that market conditions were already volatile, contributing to the spike.
- **Gazprom Pricing Revision Request**: A request for pricing revision from Gazprom in August 2021 indicates ongoing tensions and adjustments in gas pricing, which likely influenced market expectations and pricing strategies leading up to the anomaly.
- **Speculative Rally in European Gas Prices**: The speculative rally noted in December 2021 may have set a precedent for heightened price volatility, impacting the market dynamics in the months leading to September 2022.

2. **Connection of Temporal Events, Entities, and Themes**
- The **energy price surge** event on September 1, 2022, is the most direct temporal event linked to the anomaly. It highlights the immediate market conditions that led to the price spike.
- The involvement of **Gazprom**, a major player in the natural gas market, is crucial. Their pricing strategies and market behavior significantly affect energy prices across Europe, including Greece.
- The **European Union's** energy policies and market dynamics, as reflected in the various entities and events, also play a role in shaping the energy landscape, contributing to price fluctuations.

3. **Understanding the Context**
- The context reveals a backdrop of increasing energy prices influenced by geopolitical events, such as the ongoing tensions related to the Russian Federation and its impact on gas supplies to Europe. The annexation of Crimea in 2014 and subsequent events have created a climate of uncertainty in energy markets.
- The temporal events indicate a pattern of rising energy prices due to both market speculation and actual supply constraints, which culminated in the observed anomaly.

4. **Concise, Evidence-Based Explanation**
The anomaly in Greece's day-ahead energy price on September 1, 2022, coincides with a significant energy price surge on the same day, indicating a direct correlation. Contributing factors include ongoing pricing revisions by Gazprom and a speculative rally in European gas prices from previous months. The geopolitical landscape, particularly tensions involving the Russian Federation, has created a volatile energy market, leading to the observed spike in prices. This context underscores the interconnectedness of geopolitical events, market speculation, and energy pricing dynamics in Europe.

Figure 5-13: Which primary evidence coincide with the anomaly?-GraphRAG response.

2. *Question to GraphRAG: Which entities are involved to the price peak?*

   GraphRAG Response:

💡 **Answer:**

Explanation of Price Peak Anomaly in Greece

1. **Most Likely Causes of the Anomaly**:
   The price peak observed on September 1, 2022, with a day-ahead price of approximately €597.02, can be attributed to several interconnected factors:
   - **Geopolitical Tensions**: The ongoing conflict involving Ukraine and the Russian Federation has significantly impacted energy supply chains and pricing. The annexation of Crimea in 2014 and subsequent tensions have led to instability in energy markets.
   - **Natural Gas Pricing**: The surge in natural gas prices, particularly influenced by Gazprom's pricing strategies and the speculative rally in European gas prices, has directly affected electricity prices in Greece.
   - **Market Dynamics**: The lack of competition among electricity companies in Greece, as noted in various articles, may have exacerbated the price increases during this period.

2. **Connection of Temporal Events, Entities, and Themes**:
   - **Temporal Events**: The energy price surge noted on September 1, 2021, and the Gazprom pricing revision request in August 2021 are critical events leading up to the anomaly. These events highlight the volatility in energy prices and the influence of major suppliers like Gazprom.
   - **Entities Involved**: Key entities include:
     - **Gazprom**: A major player in the natural gas market, influencing prices through supply decisions.
     - **Greece**: The country experiencing the price spike, affected by both local market conditions and international energy dynamics.
     - **EU and IEA**: Organizations that monitor and respond to energy market conditions, potentially influencing policy and market stability.
   - **Themes**: The overarching theme of energy price volatility is evident, with numerous events linked to geopolitical tensions and market responses.

3. **Contextual Understanding**:
   The context provided indicates a complex interplay of geopolitical events, market dynamics, and energy supply issues. The historical events, such as the OPEC oil embargo and the annexation of Crimea, set a precedent for how geopolitical tensions can lead to energy price spikes. The involvement of major firms like Gazprom and the lack of competition in the Greek market further complicate the situation, leading to significant price fluctuations.

4. **Concise, Evidence-Based Explanation**:
   The price peak in Greece on September 1, 2022, can be attributed to a combination of geopolitical tensions, particularly related to the Russian Federation and Ukraine, and the resulting volatility in natural gas prices driven by Gazprom's market strategies. The lack of competition among Greek electricity suppliers has further exacerbated the situation, leading to a significant rise in day-ahead prices. This anomaly reflects broader trends in energy markets influenced by historical events and current geopolitical dynamics.

Figure 5-14: Which entities are involved to the price peak?-GraphRAG response.

Summary:

The GraphRAG-generated explanation demonstrated a high degree of factual and contextual accuracy, effectively linking the identified anomaly in Greece's day-ahead electricity price (September 2022) to relevant geopolitical and market events. The model correctly highlighted the Russian–Ukrainian conflict, Gazprom's pricing actions, and European energy policy discussions as dominant causal themes—elements that are well-documented in the historical record. Each cited entity and event could be traced to a valid source node in the knowledge graph, confirming the faithfulness and provenance integrity of the system. Minor inconsistencies appeared in temporal granularity and event grouping, reflecting residual noise in article extraction rather than reasoning errors. Overall, the response captured the multi-factor

nature of the anomaly with minimal hallucination, maintaining semantic coherence and alignment between retrieved evidence and narrative synthesis. This outcome validates the GraphRAG framework as a credible and audit-ready approach for explainable energy-data interpretation.

CHAPTER
6

6. CONCLUSIONS & FUTURE RESEARCH

6.1. Key findings

This thesis proposes a comprehensive approach to managing spatial references across open energy-market and infrastructure datasets. From the evaluation results, comparisons with alternative approaches, and the case studies, several key findings emerge: **Explainability and documentation:** The proposed system delivers valid, well-documented explanations for anomalies in time series. Each generated explanation of an abnormal phenomenon is accompanied by references to real-world events and data sources (provenance), providing transparency to the end user. In contrast to approaches that rely exclusively on LLMs and may produce unsubstantiated information, the GraphRAG methodology ensured that the model did not exhibit "hallucinations"—that is, every claim could be traced back to a node in the knowledge graph or to a recorded source. This feature strengthens trust in the outputs and facilitates interpretability, addressing a critical gap in traditional "black-box" models.

- **Reproducibility of results:** he proposed architecture and methodology emphasize reproducibility. All data sources are open and documented, and the processing pipeline is fully specified and automated, making experiments easy to repeat. Using a knowledge graph with a clearly defined ontology, along with versioned datasets and fixed random seeds where needed, ensures that other parties can verify the results. Consequently, each conclusion rests on systematically controlled data and metrics, which enhances the reliability and repeatability of the research.

- **Architectural benefits and flexibility:** The implemented multi-layer architecture (Data Acquisition → Semantic Extraction → Knowledge Graph → Applications)

demonstrated the value of clear separation of concerns and the integration of heterogeneous technologies. Each layer can evolve or be optimized independently (e.g., replace or improve the anomaly detector, upgrade the LLM or the graph) without disrupting the others. This modular design makes the system flexible for future extensions. At the same time, coupling time-series data with semantic knowledge enabled a holistic treatment of the problem: the system not only detects when an anomaly occurs, but also explains why, bridging the gap between statistical analysis and operational insight.

- **Generalizability and validity across scenarios:** The case studies (a geopolitical crisis such as the war in Ukraine, an extreme weather event such as a heatwave, and a technical infrastructure failure such as a 400 kV transmission-line outage) showed that the proposed methodology can handle anomalies of different types. In each scenario, the system identified the relevant drivers (e.g., price movements due to geopolitical events, demand spikes due to weather, system destabilization due to failure) and provided context-appropriate explanations. This indicates a wide range of applicability that can generalize to diverse data and markets, provided the corresponding knowledge is represented in the graph. It also suggests that a globally scoped knowledge graph can capture multi-factor interactions more effectively than monolithic approaches.

6.2. Limitations

Despite these contributions and positive results, several limitations of the present work should be acknowledged:

- **Limited data coverage and generalization:** The platform was developed and evaluated primarily on data from the Greek electricity market and English-language news sources. Geographic and linguistic coverage is therefore limited. Some events or anomaly drivers may be missing from the knowledge graph if they are not present in the considered sources. As a result, the generalization of the conclusions to other markets or regions (especially those with different languages or regulatory settings) has not been fully verified. Adapting the system (ontologies, data sources, models) is required to extend coverage internationally.

- **Ontology alignment and extraction accuracy:** Mapping extracted entities and events to a predefined ontology poses challenges. Although techniques such as ontology-aware prompting and JSON-schema checks were applied, there is no guarantee that all extracted information aligns perfectly with the target knowledge model. Mismatches or misclassifications can occur—e.g., novel or unusual event types may be categorized incorrectly—reducing homogeneity and graph quality. Moreover, LLM-based event extraction is not infallible: an LLM may occasionally fail to capture an event or assign it to the wrong class, necessitating additional verification mechanisms and human-in-the-loop correction for critical cases.

- **Prompt/LLM stability:** ΠEven with prompt optimization, large language models can exhibit output variability. Small prompt changes or model updates may alter responses. An LLM can also circumvent constraints or behave unpredictably when operating outside its trained domain. Although no obvious hallucinations were observed here thanks to the RAG mechanism, the risk persists under different conditions. Continuous monitoring and retuning are practical necessities. In addition, reliance on a specific LLM (e.g., a large model such as GPT-4) creates dependence on the provider's platform and constraints (API costs, rate limits, and ethical/legal usage considerations).

- **Computational cost and scale:** Integrating a knowledge graph and an LLM introduces complexity and computational overhead. Graph queries (e.g., complex Cypher over a large Neo4j) and LLM calls add latency. While current response times (~2 seconds for medium-complexity queries) were acceptable, scaling may be challenging. With orders-of-magnitude more nodes/relationships or many concurrent users, performance could degrade. Running a large LLM in real time is also computationally (or financially) expensive when via a cloud API. The system has not yet been optimized for large-scale production, and continuous ingestion/updating of the graph (streaming) was not fully implemented—limiting real-time expansion of the knowledge base.

6.3. Future directions

The above limitations and findings open multiple avenues for further research and improvement:

- **Causality-aware retrieval:** Incorporate mechanisms that reason about causality during graph retrieval. Rather than relying solely on correlations or keyword co-occurrence, employ causal-inference ideas to distinguish true cause-and-effect relations. For example, retrieval could prioritize event paths connected by CAUSES or AFFECTS-like semantics, yielding explanations that not only list relevant events but also substantiate that they caused the observed anomaly. Combining knowledge graphs with causal models should improve reliability and address the correlation-vs-causation challenge.

- **Real-time data processing:** or operational deployment (e.g., a TSO or market platform), real-time integration is critical. Future work should focus on streaming pipelines (e.g., Apache Kafka) and rolling-window anomaly detection that update the knowledge graph dynamically. The challenge is to keep latency low so that, as soon as an anomaly occurs, the system can immediately retrieve context and produce explanations. Methods for online graph updates will also be needed to maintain data consistency during continuous ingestion.

- **Multimodal prediction and analysis:** Energy markets are global. A natural next step is to support multiple languages and regions. While the current implementation leans on English material, critical events may first appear in other languages. Future work can integrate multilingual LLMs or machine translation so extraction performs effectively in Greek, English, French, and beyond. Cultural and regulatory differences should also be reflected: the graph can be enriched with local ontologies and specialized nodes (e.g., national regulators, local operators). This will increase generality and enable worldwide deployment.

- **Multilingual and cross-regional generalization:** Energy markets are global. A natural next step is to support multiple languages and regions. While the current implementation leans on English material, critical events may first appear in other languages. Future work can integrate multilingual LLMs or machine translation so extraction performs effectively in Greek, English, French, and beyond. Cultural and regulatory differences should also be reflected: the graph can be enriched with local ontologies and specialized nodes (e.g., national regulators, local operators). This will increase generality and enable worldwide deployment.

- **Architecture optimization and scaling:** Production use will require improvements in performance and resilience: more efficient graph-query strategies (smart caching, precomputed associations), lighter or specialized on-prem LLMs to reduce cost, and better scalability via distributed processing. Techniques such as AutoML/AutoPrompt can help the system self-improve over time based on fresh data and user feedback. These steps will move the platform toward a mature operational tool, ready to support energy-market stakeholders with accuracy, speed, and reliability.

In summary, the conclusions and future directions in this chapter underscore a dual contribution: on the one hand, a novel solution that bridges statistical time-series analysis with semantic knowledge in the energy domain; on the other, a roadmap for scaling and adoption. Implementing these extensions can transform the platform into a generalized, intelligent decision-support system for energy markets and infrastructure, improving understanding and management of the complex spatiotemporal dynamics that govern them.

BIBLIOGRAPHY

[1] A. Li, Y. Zhang, Z. Chen, L. Song, and W. Chen, “Large Language Models for Generative Information Extraction: A Survey,” *arXiv preprint* arXiv:2312.17617, 2023. [Online]. Available: https://arxiv.org/abs/2312.17617

[2] S. Wang, Y. Li, M. Dong, H. Liu, Y. Zhang, and M. Zhou, “From Local to Global: A Graph RAG Approach to Query-Focused Summarization,” *Microsoft Research*, 2024. [Online]. Available: https://www.microsoft.com/en-us/research/publication/from-local-to-global-a-graph-rag-approach-to-query-focused-summarization/

[3] D. Xu *et al.*, “Large Language Models for Generative Information Extraction: A Survey,” *Frontiers of Computer Science*, 2024, doi:10.1007/s11704-024-40555-y.

[4] X. Geng *et al.*, “Improving LMs with Grammar-Constrained Decoding,” in *Proc. ACL*, 2023.

[5] E. Karataş, “Structured Output Generation in LLMs: JSON Schema and Grammar-Based Decoding,” *Medium*, Feb. 15 2025. [Online]. Available: https://medium.com

[6] A. Edge *et al.*, “GraphRAG: Unlocking LLMs for Retrieval on Large Document Collections,” *Microsoft Research*, 2024. [Online]. Available: https://microsoft.github.io/GraphRAG/

[7] Neo4j, “Neo4j Graph Database Platform — Developer Guides & Cypher,” 2024–2025. [Online]. Available: https://neo4j.com/docs/

[8] LangChain, “Graph RAG & Neo4j Integrations (Neo4jGraph, GraphCypherQAChain, Neo4jVector),” 2024–2025. [Online]. Available: https://blog.langchain.com/

[9] ERCIM News, “GLACIATION: Distributed Knowledge Graphs for Energy Systems (SCADA-Driven Anomaly Detection),” 2024. [Online]. Available: https://ercim-news.ercim.eu/

[10] Neo4j, “Utilities Are Graphs: Using Neo4j for Energy Networks,” 2024–2025. [Online]. Available: https://neo4j.com/

[11] Neptune.ai, “Data / Model Drift: Monitoring Best Practices,” 2024–2025. [Online]. Available: https://neptune.ai/

[12] OTexts, “Forecasting: Principles and Practice — Rolling Forecast Origin,” *Monash University*, Online. [Online]. Available: https://otexts.com/

[13] Machine Learning Mastery, “Walk-Forward Validation for Time Series Forecasting,” Online. [Online]. Available: https://machinelearningmastery.com/