Schema Discovery on Property Graphs with Large Language Models

A Training-Free, Dataset-Agnostic Pipeline with Ground-Truth Evaluation

George Souladakis, Sylvana Kakarontza, Marios Papadakis, and Marsel Senka

Computer Science Department, University of Crete

Abstract. Property graphs are widely used to represent complex and heterogeneous data, yet many real-world graph datasets lack explicit schemas, hindering understanding, querying, and validation. Automatically discovering schemas for large property graphs is challenging due to scale, noise, and diverse data representations. In this work, we propose a hybrid approach for property graph schema discovery that combines scalable statistical profiling, deterministic structural heuristics, and large language model reasoning. Our system performs streaming analysis over raw graph data to extract type statistics, property distributions, and topological patterns without materializing the full graph. These structured profiles are provided to a large language model (Gemini), which infers a high-level schema describing node types, edge types, properties, and connectivity constraints. The inferred schema is then normalized and systematically compared against an available ground-truth schema to evaluate structural and semantic correctness. This evaluation demonstrates that the proposed approach can recover accurate and interpretable schemas that closely reflect the underlying data, highlighting the effectiveness of integrating symbolic graph profiling with large language models for scalable schema discovery.

Keywords: Schema Discovery · Property Graphs · Large Language Models · Graph Profiling · Ground-Truth Evaluation

1 Introduction

Property graphs have become a widely adopted data model for representing complex and heterogeneous relationships in domains such as social networks, knowledge graphs, biological systems, and information integration platforms. Modern graph database systems and analytics pipelines rely heavily on property graphs due to their expressive power, allowing both nodes and edges to carry labels and rich sets of properties. Despite their widespread use, many real-world property graph datasets lack explicit or up-to-date schema definitions, making it difficult for users and systems to understand the structure of the data, formulate correct queries, and perform validation or integration tasks.

Schema discovery aims to automatically infer a high-level structural description of a graph, including node types, edge types, property sets, and connectivity constraints, directly from the data. However, discovering schemas for large property graphs remains a challenging problem. Graph datasets are often noisy, heterogeneous, and distributed across multiple files or sources, with inconsistent naming conventions and incomplete metadata. Traditional schema inference approaches typically rely on manual inspection, rigid rule-based techniques, or purely statistical methods, which struggle to scale and often fail to capture higher-level semantic structure.

Recent advances in large language models (LLMs) have shown promise in extracting structure and meaning from unstructured and semi-structured data. Their ability to reason over textual descriptions suggests that they could play a role in schema discovery. However, directly applying LLMs to raw graph data is impractical for large graphs due to scale limitations and the lack of explicit structural representation in the input. Moreover, unconstrained LLM inference may introduce hallucinations or over-generalizations that do not faithfully reflect the underlying data.

In this work, we present a hybrid approach for property graph schema discovery that combines scalable data profiling, deterministic structural heuristics, and LLM-based reasoning. Instead of exposing raw graph data to the language model, our system first performs a streaming analysis over the input datasets to extract compact yet informative summaries, including type statistics, property distributions, and observed topological patterns. These structured profiles serve as a controlled and interpretable representation of the graph and are provided as input to a large language model (Gemini), which infers a candidate schema.

To ensure robustness and correctness, the inferred schema is further normalized and post-processed using deterministic rules, and its quality is evaluated through systematic comparison against an available ground-truth schema. This design enables scalable and dataset-agnostic schema discovery while maintaining interpretability and evaluability. Overall, our approach demonstrates how symbolic graph profiling and large language models can be effectively combined to address the challenges of schema discovery in large property graphs.

2 Related Work

Schema discovery for property graphs has been studied as graph databases have become increasingly adopted in practice. Prior work has primarily focused on defining formal property graph models and schema representations, highlighting the need for explicit schemas to support querying, validation, and data understanding [1]. While these efforts establish the foundations of property graph schemas, they do not address automatic schema inference from data.

Several approaches have explored schema discovery for property graphs using statistical aggregation and heuristic analysis, extracting node and edge labels, property keys, and simple connectivity patterns from graph instances. Profiling-based techniques improve scalability by summarizing large graphs at the type level, but they rely on fixed heuristics and offer limited semantic abstraction [2].

More recently, large language models (LLMs) have been investigated for schema inference and structural reasoning tasks. These approaches demonstrate that LLMs can infer schema-like abstractions when provided with structured summaries rather than raw data. However, existing work typically emphasizes qualitative analysis and lacks systematic evaluation against ground-truth property graph schemas [4].

To support quantitative evaluation, recent benchmark efforts provide standardized property graph datasets together with reference schemas [3]. Our work builds on these benchmarks and differs from prior approaches by combining deterministic property graph profiling with constrained LLM-based inference, followed by normalization and quantitative comparison against ground-truth schemas.

3 System Overview

This section provides an overview of the proposed schema discovery system, highlighting its main components and the data flow between them. The system is designed to infer property graph schemas from raw input data in a scalable, dataset-agnostic, and interpretable manner. Figure **??** illustrates the high-level architecture of the pipeline.

3.1 Pipeline Architecture

The schema discovery process consists of four main stages: data ingestion and profiling, structured summary construction, LLM-based schema inference, and post-processing and evaluation. The pipeline operates on graph data stored in tabular form, where nodes and edges are typically provided as separate files with associated identifiers and properties.

In the first stage, the system performs streaming data ingestion and profiling. Input files are automatically classified as node or edge files based on their column structure. Rather than materializing the full graph in memory, the system processes the data in chunks and collects lightweight statistics, including node and edge counts, property occurrence frequencies, inferred property types, and observed connectivity patterns between node types. This design allows the system to scale to large datasets while remaining robust to noisy or heterogeneous input formats.

3.2 Structured Graph Profiling

The collected statistics are aggregated into structured type-level summaries that describe each detected node type and edge type. For node types, the summaries capture property coverage, dominant data types, and indicative value samples. For edge types, the summaries include property presence, connectivity patterns between source and target node types, and frequency information. Additional structural heuristics are applied to identify technical or intermediate node types and to detect frequent multi-hop patterns that may indicate higher-level logical relationships.

These structured summaries serve two purposes. First, they provide a compact and interpretable representation of the graph that can be inspected independently of the inference process. Second, they act as a controlled interface between the raw data and the language model, ensuring that the LLM operates on distilled structural information rather than unbounded raw input.

3.3 LLM-Based Schema Inference

The structured profiles are converted into a textual representation and provided as input to a large language model (Gemini). The model is instructed to infer a property graph schema that reflects the observed structure of the data, including node types, edge types, property definitions, and connectivity constraints. The inference process is guided by explicit constraints that discourage uncontrolled merging of types and require the model to preserve distinct structural elements present in the input profiles.

By delegating high-level schema reasoning to the language model while constraining its input and output format, the system leverages the model's semantic capabilities without sacrificing structural fidelity or scalability.

3.4 Post-Processing and Evaluation

The schema produced by the language model is further refined through deterministic post-processing steps. These steps normalize naming conventions, complete missing property definitions using profiling statistics, and ensure internal consistency of the schema. Finally, the inferred schema is compared against an available ground-truth schema using a dedicated evaluation module. This comparison assesses the correctness of inferred node and edge types, property sets, and topological constraints, enabling systematic analysis of the system's performance.

Together, these stages form an end-to-end pipeline for property graph schema discovery that combines scalable data analysis, symbolic reasoning, and large language model inference.

4 Data Profiling and Statistics

A key design principle of the proposed system is the extraction of informative structural signals from graph data without materializing the full graph in memory. To achieve this, the system employs a streaming data profiling strategy that operates directly on the input files and produces compact, type-level summaries suitable for downstream reasoning and schema inference.

4.1 Streaming Data Ingestion

Graph datasets are provided as collections of tabular files, typically representing node and edge instances. During ingestion, each file is automatically classified as either a node file or an edge file based on its column structure, including the detection of identifier columns and source–target relationships. This automatic role detection allows the system to operate in a dataset-agnostic manner and eliminates the need for dataset-specific configuration.

To support large-scale datasets, files are processed incrementally in chunks. This streaming approach enables the system to handle graphs that exceed main memory capacity while maintaining stable performance and predictable resource usage.

4.2 Node Type Profiling

For each detected node type, the system collects statistical summaries that characterize its structure and properties. These summaries include the total number of instances, property occurrence frequencies, inferred property data types, and representative value samples. Property data types are inferred using lightweight type detection heuristics based on observed values rather than schema assumptions.

The profiling process records how frequently each property appears across instances of a node type, allowing the system to distinguish between mandatory and optional properties. This information is later used both to guide schema inference and to support deterministic post-processing of inferred schemas.

4.3 Edge Type Profiling

Edge profiling focuses on capturing both property-level and topological information. For each edge type, the system records the frequency of observed properties as well as the distribution of source and target node type pairs. These connectivity patterns provide essential signals about the allowed relationships between node types and form the basis for topological constraints in the inferred schema.

To enable scalable topology analysis, the system maintains a lightweight mapping from node identifiers to node types, allowing edge instances to be resolved to type-level connections without constructing the full graph. This design enables efficient aggregation of topological patterns even for large edge sets.

4.4 Structural Heuristics

Beyond basic statistics, the system applies deterministic structural heuristics to enrich the profiling results. These heuristics identify node types that act as technical or intermediate containers, such as join-like nodes with sparse properties and multiple incoming and outgoing connections. In addition, the system analyzes frequent multi-hop patterns that suggest higher-level logical relationships between node types.

The outputs of these heuristics are incorporated into the profiling summaries and exposed to the language model as structured signals rather than hard constraints. This design allows the model to reason about potential abstractions while remaining grounded in observable data characteristics.

Overall, the data profiling stage produces a concise yet expressive representation of the underlying graph structure, balancing scalability with semantic richness and serving as the foundation for subsequent schema inference.

5 LLM-Based Schema Inference

The schema inference stage leverages a large language model to perform high-level structural reasoning based on the profiled representation of the graph. Rather than operating on raw data, the model is guided by compact, structured summaries that capture the essential characteristics of node types, edge types, properties, and connectivity patterns observed during profiling.

5.1 Profile-to-Text Transformation

The statistical summaries produced during data profiling are converted into a textual representation that preserves structural information while remaining concise. For each node type, the profile describes the number of instances, observed properties, their relative coverage, inferred data types, and indicative value patterns. For each edge type, the profile includes observed source–target type pairs, frequency information, and property presence.

In addition, the profiling stage may generate higher-level structural observations, such as frequently occurring multi-hop patterns or intermediate node types that appear to act as technical connectors. These observations are included as contextual information rather than enforced constraints, allowing the language model to incorporate them during inference when appropriate.

5.2 Prompt Design and Constraints

The language model is instructed to infer a property graph schema that faithfully reflects the structure described in the input profiles. The prompt explicitly constrains the model to preserve distinct node and edge types, avoid uncontrolled merging of structurally different elements, and exclude implementation-specific artifacts such as import-related columns or internal identifiers.

The schema is required to follow a fixed JSON format that specifies node types, edge types, property definitions, and connectivity information. By enforcing a strict output structure, the system ensures that the model's output can be parsed, normalized, and evaluated deterministically.

5.3 Schema Generation

Given the structured profile and the constrained prompt, the language model produces a candidate schema describing the inferred node and edge types, their properties, and the relationships between them. The model is responsible for synthesizing a coherent schema that integrates statistical signals, structural patterns, and semantic cues present in the profiling summaries.

By separating low-level statistical extraction from high-level reasoning, the system leverages the strengths of large language models—namely abstraction and semantic generalization—while minimizing exposure to noise and scale-related limitations. This hybrid design allows the inference process to remain both scalable and grounded in observable data characteristics.

6 Schema Normalization and Post-Processing

The raw schema produced by the language model may contain inconsistencies, incomplete property definitions, or variations in naming that arise from probabilistic inference. To ensure that the final schema is internally consistent, interpretable, and suitable for evaluation, the system applies a series of deterministic normalization and post-processing steps.

6.1 Name and Structure Normalization

All node and edge type names are normalized to enforce consistent formatting and to remove dataset-specific artifacts. This includes stripping technical prefixes, resolving minor syntactic variations, and ensuring that each type is represented by a unique canonical name. Connectivity information is normalized without collapsing distinct structural patterns, preserving all observed source–target relationships inferred from the data.

This normalization stage is deliberately conservative: it avoids semantic merging of node or edge types and focuses solely on syntactic consistency. As a result, the structural diversity captured during profiling and inference is retained in the final schema.

6.2 Property Completion and Backfilling

In some cases, the language model may omit property definitions or produce incomplete property sets, particularly for large or sparse datasets. To address this, the system deterministically backfills missing properties using the statistics collected during the profiling stage. For each node or edge type, properties observed in the data are added to the schema when absent, along with inferred data types and mandatory flags derived from property coverage ratios.

This backfilling process ensures that the inferred schema remains faithful to the underlying data, even when the model output is partially incomplete. Importantly, property completion is driven entirely by observed statistics and does not introduce new properties beyond those present in the input data.

6.3 Consistency Enforcement

Additional post-processing steps enforce internal consistency across the schema. When multiple schema elements share the same edge label, property definitions are propagated to ensure uniformity. Reserved or implementation-specific attributes, such as import-related identifiers, are removed to prevent leakage of low-level artifacts into the schema definition.

Through these normalization and post-processing steps, the system transforms the model-generated schema into a stable and reproducible representation that can be directly compared against ground-truth schemas and used for downstream tasks.

7 Evaluation

This section evaluates the effectiveness of the proposed schema discovery pipeline by comparing inferred schemas against available ground-truth schemas. The evaluation is designed to assess both the structural correctness and the property-level accuracy of the inferred schemas, while avoiding dataset-specific tuning or manual intervention.

7.1 Datasets and Ground Truth

We evaluate our approach on multiple property graph datasets for which ground-truth schema definitions are available. Each dataset consists of node and edge data provided in tabular form, along with a corresponding reference schema describing node types, edge types, property sets, and connectivity constraints.

For each dataset, the ground-truth schema is extracted and normalized using a dedicated parsing module to ensure consistency with the inferred schema representation. This normalization step aligns naming conventions and structural formats, enabling fair and systematic comparison between inferred and reference schemas.

7.2 Evaluation Protocol

For each dataset, the schema discovery pipeline is executed once to produce a single inferred schema. This schema is generated using deterministic profiling and post-processing steps combined with a constrained large language model inference. The resulting inferred schema is then compared against the corresponding ground-truth schema using a fixed evaluation procedure.

We deliberately avoid repeated inference runs or prompt tuning during the main evaluation in order to reflect a realistic, fully automated usage scenario. This protocol ensures that reported results correspond to a single end-to-end execution of the system per dataset.

7.3 Evaluation Metrics

The comparison between inferred and ground-truth schemas is performed at multiple levels of granularity. We report the following metrics:

– **Node Type Accuracy:** Measures the proportion of ground-truth node types that are correctly identified in the inferred schema.
– **Edge Type Correctness:** Measures the proportion of ground-truth edge types that are correctly recovered, taking into account edge labels and associated source–target type constraints.
– **Node Property Accuracy:** Measures the overlap between property sets defined for each node type in the inferred and ground-truth schemas.
– **Edge Property Accuracy:** Measures the overlap between property sets defined for each edge type in the inferred and ground-truth schemas.
– **Topology Accuracy:** Evaluates the correctness of inferred connectivity constraints by comparing allowed source–target type pairs between inferred and ground-truth schemas.

Each metric is computed using exact or normalized matching of schema elements after post-processing, ensuring that differences due solely to syntactic variations do not affect the evaluation.

7.4 Datasets and Ground Truth

The datasets used in this evaluation are obtained from a publicly available property graph benchmark suite that provides both graph data and corresponding ground-truth schemas [3]. The benchmark includes a collection of heterogeneous property graph datasets of varying size and structural complexity, enabling systematic and reproducible evaluation of schema discovery techniques.

7.5 Schema Matching and Comparison

Schema elements in the inferred and ground-truth schemas are matched using normalized type names and structural information. Node and edge types are aligned based on their canonical names, while property sets are compared using set-based matching. For edge types, both label matching and topology constraints are considered when determining correctness.

The comparison process is fully automated and deterministic. No manual alignment or dataset-specific rules are applied during evaluation, ensuring that results reflect the intrinsic performance of the schema discovery pipeline.

7.6 Results

Table 1 summarizes the evaluation results across the tested datasets. For each dataset, we report the metrics described above, along with an overall performance score that aggregates type-, property-, and topology-level correctness.

Table 1. Schema discovery accuracy percentages results across datasets.

Dataset	Node	Edge.	Node Props	Edge Props	F1.	Overall
Starwars	100	90	100	N/A	94.74	96.18
Pole	100	76.47	100	100	83.87	95.83
MB6	60	40	85	100	44.44	65.89
FIB25	60	60	83	100	47.15	69.90
LDBC	100	82.61	93.55	100	88.37	92.91

7.7 Robustness Analysis

Although the primary evaluation uses a single inference run per dataset, we additionally assess the robustness of the approach with respect to the stochastic nature of large language model inference. For a representative dataset, we repeat the schema inference process multiple times using identical profiling input and prompt structure.

We observe that the inferred schemas remain largely consistent across runs, with only minor variations in naming or property grouping. Quantitative differences across runs are reported using mean and standard deviation of the evaluation metrics. This analysis indicates that the combination of structured profiling, constrained prompting, and deterministic post-processing significantly reduces variability in the inferred schemas.

7.8 Discussion of Errors

To better understand the limitations of the proposed approach, we qualitatively analyze cases where the inferred schema deviates from the ground truth. Common sources of error include ambiguous type naming, sparsely populated properties, and edge types with highly heterogeneous connectivity patterns. These observations highlight directions for future improvements, such as enhanced disambiguation heuristics and richer structural summaries.

A LLM Prompt for Schema Inference

The following prompt is used to guide the large language model during schema inference. The prompt operates on structured graph profiling summaries and enforces a fixed output format.

```
You are a Senior Property Graph Schema Architect. Your mission is to infer a high-fidelity
   DATA PROFILE:
   <serialized type-level summaries generated by the profiling stage>


   TARGET:
   The user needs a schema that preserves 100% of the nodes and edge properties found in t
```

```
 CRITICAL AGNOSTIC HEURISTICS (Apply strictly in this order):

 1. **NO "SMART" MERGING (Distinctness Rule) - PRIORITY #1:**
    - **Constraint:** If the Data Profile lists distinct node types (e.g., "EntityA" and
    - **Reasoning:** Even if they share properties, they represent different entities.
    - **Strict Instruction:** Do NOT merge nodes just because they look similar. Keep the

 2. **NOISE FILTER (The "Fake Node" Check):**
    - **Goal:** Identify properties that masquerade as nodes (e.g., Tags, Categories, Lab
    - **Detection Logic:** Look at the [STRUCTURAL FINGERPRINTS] section. A node is a "Fa
      * **Criteria A:** Low Information Density (Avg Properties < 2, excluding purely int
      * **Criteria B:** Passive Role (It is a "Leaf" or "Sink" node with 0 outgoing edges
    - **Action:** If detected, DELETE the Node Type and add its name as a property to the
    - **EXCEPTION:** If the node has *outgoing edges* to other entities, it is a structur

 3. **PROPERTY FORMATTING (Strict JSON Structure):**
    - **Constraint:** The 'name' field in your JSON must contain **ONLY the property name
    - **Forbidden:** Do NOT include the type in the name (e.g., "id:long" is WRONG).
    - **Correct:** "name": "id", "type": "Long".
    - **Forbidden:** Do NOT output internal Neo4j keys like ":START_ID" or ":END_ID".

 4. EDGE NAMING METHODOLOGY (SEMANTIC DERIVATION):
    - **Constraint:** Do not use a pre-set list of verbs. Deriving the name must follow t

    * **STEP 1: ANALYZE SIGNAL:** Look at the edge properties and the Source/Target types
        * *Signal A:* Properties imply measurement (weight, distance, score).
        * *Signal B:* Properties imply sequence or action (time, duration, flow).
        * *Signal C:* Relationship implies ownership or composition (part-of, member-of)

    * **STEP 2: DETERMINE CATEGORY:**
        * If *Signal A* (Measurement) -> The Category is **TOPOLOGICAL**. Use verbs descr
        * If *Signal B* (Action) -> The Category is **FUNCTIONAL**. Use specific active v
        * If *Signal C* (Ownership) -> The Category is **STRUCTURAL**. Use verbs describi

    * **STEP 3: GRAMMAR FILTER (REDUNDANCY REMOVAL):**
        * **Rule:** The Edge Name MUST NOT repeat the Target Node's name.
        * *Bad:* `Parent` -> `HAS_PARENT_GROUP` -> `Group` (Redundant).
        * *Good:* `Parent` -> `INCLUDES` -> `Group`.
        * *Bad:* `System` -> `LINKS_TO_SYSTEM` -> `System` (Redundant).
        * *Good:* `System` -> `CONNECTS` -> `System` (if topological) or `INTERACTS` ->

OUTPUT JSON FORMAT:
 {{
   "node_types": [
     {{
          "name": "NodeLabel",
          "properties": [
            {{"name": "propertyName", "type": "String|Long|Double|Boolean|StringArray|Point
          ]
        }}
      ],
      "edge_types": [
        {{
          "name": "INFERRED_SEMANTIC_VERB",
          "start_node": "SourceNodeLabel",
          "end_node": "TargetNodeLabel",
          "properties": [
            {{"name": "propertyName", "type": "String|Long|Double|Boolean", "mandatory": tru
          ]
        }}
      ]
    }}
```

References

1. Angles, R.: The property graph database model. In: Proceedings of the 13th Alberto Mendelzon International Workshop on Foundations of Data Management (AMW) (2018), http://ceur-ws.org/Vol-2100/
2. Bonifati, A., et al.: Graph summarization. ACM SIGMOD Record **47**(1), 40–49 (2018)
3. Sofia Sideri, Georgia Troullinou, E.Y.V.E.D.P.H.K.: Property graph benchmark datasets with ground-truth schemas (2024). https://doi.org/10.5281/zenodo.17801336, https://zenodo.org/records/17801336
4. Sofia Sideri, Georgia Troullinou, E.Y.V.E.D.P.H.K.: Large language models for schema and structure discovery. arXiv preprint arXiv:2512.01092 (2025), https://arxiv.org/abs/2512.01092