

Entity InfoCards from Heterogeneous Data Lakes

Saba Shahsavari

Inria & Institut Polytechnique de Paris

September, 2025

Supervised by: Garima Gaur, Ioana Manolescu,
Madhulika Mohanty, Georgios Siachamis.



Outline

1. **Motivation:** Data lakes and the need for entity-centric summaries
2. **Our Approach:** InfoCard Framework – end-to-end summarization across formats
3. **Evaluation:** Implementation and results

Named Entities

What are Named Entities?

- Specific, identifiable elements: **Person**, **Location**, **Organization**, **Date**.

Why do Named Entities matter?

- They shape how we think and process information



Journalist

- Who is this **politician/activist**?
- What activities were recorded on **May 12, 2024**?



Finance/Business Analyst

- Partners, funding, and products of **company C**?
- What activities has **Account 456** been involved in?



Healthcare Professional

- Who was **Patient123**?
- What communications or reports mention **Dr. X**?



Legal Analyst:

- What incidents have been reported at **Warehouse 12**?
- What deals or contracts has **Company B** been involved in?

Named Entity Summarization

Named Entity Recognition: Detects and classifies named entities in text, but it does not tell us:

- What the entity *is* (roles, attributes)
- How it *varies across contexts*
- How it *relates to other entities*

Named Entity Summarization: Creates **concise, entity-focused summaries** that explain

- *What an entity is, what it does, and how it is related to other entities*

Named Entity Recognition

...

- Ben → Person
- California → Location
- C1 → Organization

Named Entity Summarization

Entity: Ben

- Born in California
- Founded company C1

What is a Data Lake?

- Store **vast amounts** of raw, heterogeneous data
- Direct storage without strict schema
- Flexible and adaptable to new sources
- Highly scalable for modern data growth

Challenges in querying data lakes:

- ❑ No unified schema
- ❑ Incomplete/inconsistent metadata
- ❑ No consistent query language
- ❑ Information about a single entity is scattered



InfoCard Framework

We propose a system to generate entity summaries, or *InfoCards*, from raw data.

- Handles multiple data formats
- Decouples data extraction from summarization
- Precomputed and cached for fast generation on any entity
- Ensures efficiency and scalability for large-scale data lakes

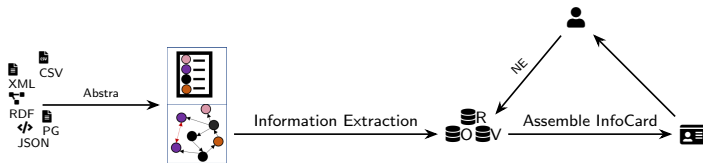
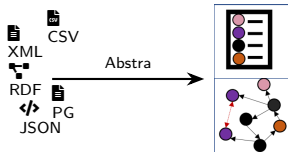


Figure: InfoCard generation framework

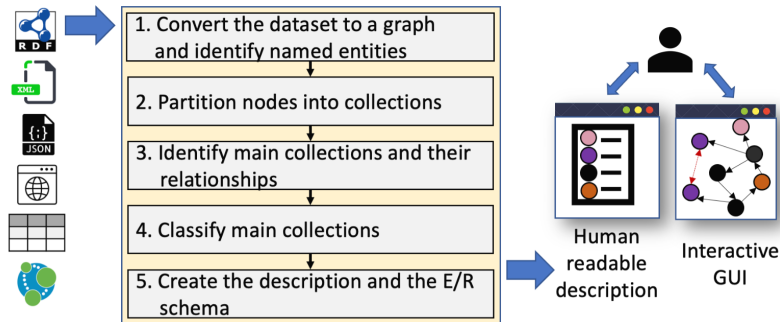
InfoCard Framework

Step 0 Integrate data into a **graph** structure



Abstra: Integrating Data Lakes into a Unified Graph

Abstra [1] integrates heterogeneous data into a graph structure.



What Does the Abstra Graph Look Like?

- Normalized structure
- Named entities only as leaf nodes
- Nodes grouped in collections

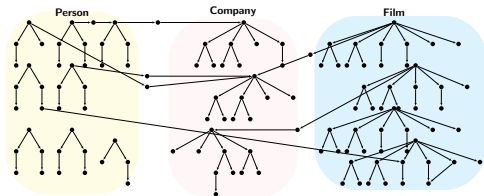


Figure: Simplified illustration of an Abstra graph (example derived from an RDF dataset)

Why not query the graph directly?

- Millions of nodes/edges
- Complex structure → slow real-time queries
- Requires custom queries



: Abstra helps us understand what the data is about

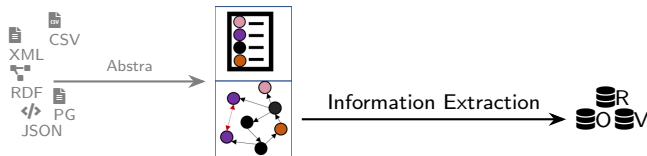


: Does not give fine-grained entity summaries

InfoCard Framework

Step 0 Integrate data into a graph structure

Step 1 Information extraction from graph data



Information extraction

Goal: Compact, query-friendly representation of Abstra's graph.

Approach: Flatten the graph into three tables:

- **Occurrence Table (O):** Stores named entity occurrences
- **Value Table (V):** Stores literal values
- **Relation Table (R):** Stores relations.

Information extraction: From graph to relational tables

How to store data in relational tables in a way that it is:

- Complete: No facts lost.
- Minimal: No redundancies.
- Efficient: Fast queries.
- Structured: Easy InfoCard assembly.

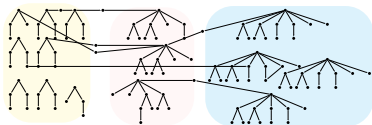


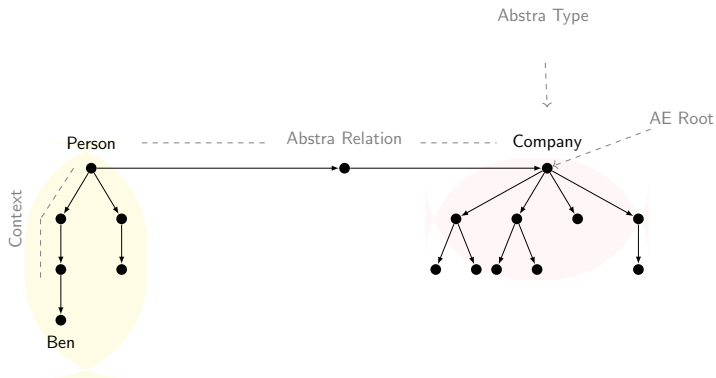
Figure: Simplified illustration of an Abstra graph

τ	a	p	i	λ	e
τ	a	p	i	v	
τ	a	τ'	d'	l	

Figure: Relational tables

Information extraction: Building Blocks

- **Abstra Entity (AE):** subgraph rooted at a collection.
- **Abstra Type (τ):** category of the AE root.
- **Context Path (p):** path from root to a named entity.
- **Abstra Relation:** A path that connects two AE roots.



Example: Encoding Graph Data into Tables

A simplified example of mapping the graph into three tables:

Value table

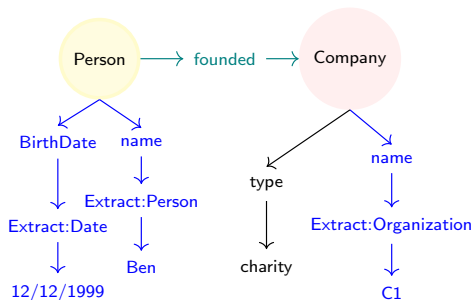
τ	a	p	i	v
Company	a2	type	type	Charity

Relation table

τ	a	τ'	a'	l
Person	a1	Company	a2	Founded

Occurrence table

τ	a	p	i	λ	e
Person	a1	name	name.http:example.fr/name.ben	Person	Ben
Person	a1	Birthdate	Birthdate	Date	12/12/1999
Company	a2	name	name	Organization	C1

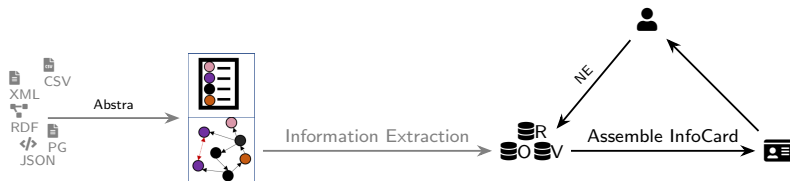


InfoCard Framework

Step 0 Integrate data into a graph structure

Step 1 Information extraction from graph data

step 2 Assemble infoCard



Querying for Named Entities

- InfoCard assembly begins by querying the occurrence table for a target named entity.

Let's search for *Ben*. This reveals the Abstra entities where *Ben* appears.

τ	a	p	i	λ	e
⋮					
Person	a1	name	name.BenURI	Person	Ben
Film	a10	director.name	director.name	Person	Ben
⋮					

Table: Occurrence Table snippet

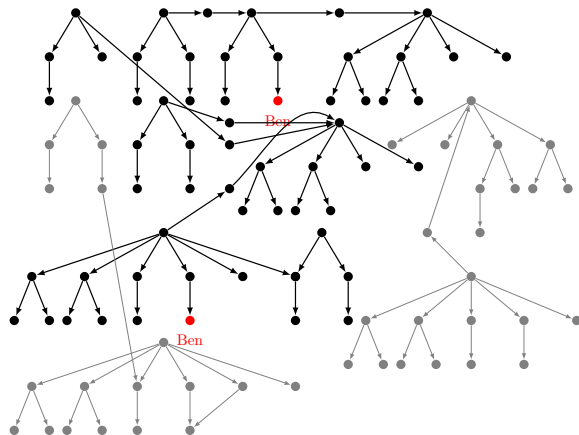
Following Ben through the data graph:

- Query the Value table for other attributes of these entities.
- Query the Relation table for connected entities.
- This may uncover new Abstra entities...
- ...leading back to Value and Occurrence tables for their attributes.

When should we stop?

Querying for Named Entities - At Graph Level

- Querying iteratively builds a subgraph around the named entity.
- **Challenge:** How to focus iteration on informative graph parts?
- **Approach:** Define and use the notion of *key context*.

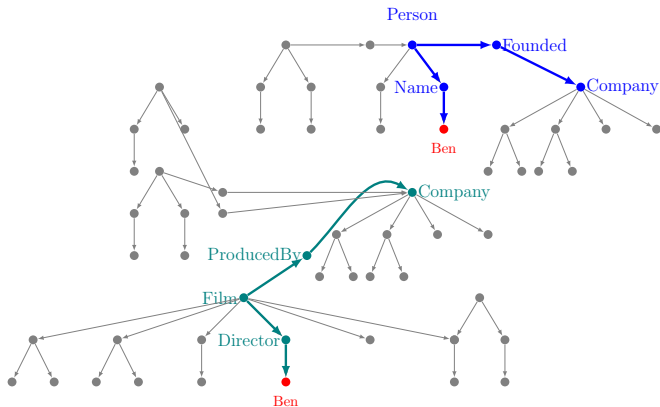


Finding the key Context

We need to find the occurrence that is most descriptive or "key" to the entity.

Question: If you could keep only one fact about *Ben*, which would it be?

1. Ben is a person who founded a company.
2. Ben is a director of a movie produced by a company.



Defining Key Context

We define the criteria that make a path 'key,' and we measure these criteria based on their deviations from an ideal case.

Criteria for a "Key" context:

- **Frequency:** Common across entities of the same type
- **Type Association:** Always leads to the same NE type
- **Consistency:** Similar occurrence patterns across entities
- **Uniqueness:** Produces distinct values across instances

Penalties to Select the Key Context:

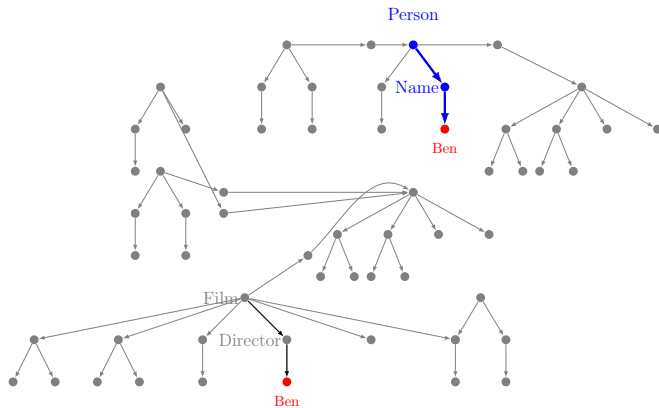
- ϵ_1 : Penalizes the fraction of AEs missing this path.
- ϵ_2 : Penalizes paths ending in NEs of different types.
- ϵ_3 : Penalizes multiple path occurrences within AEs.
- ϵ_4 : Penalizes based on the proportion of distinct instances.

Result: Lowest total penalty designates the key context.

Assembling an InfoCard

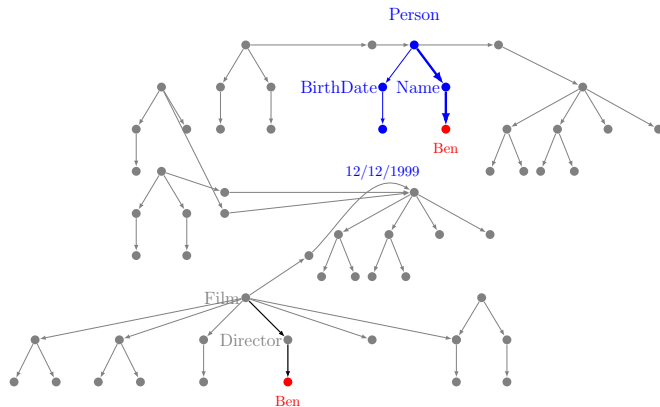
1. Find the key context.

$\text{Penalty}(\text{Person}, \text{name} \rightarrow \text{Extract} : p) < \text{penalty}(\text{Film}, \text{director} \rightarrow \text{Extract} : p)$



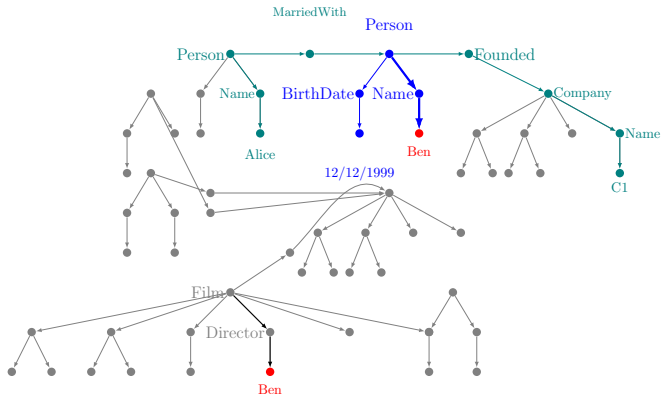
Assembling an InfoCard

1. Find the key context.
2. Retrieve all the information from the key AE



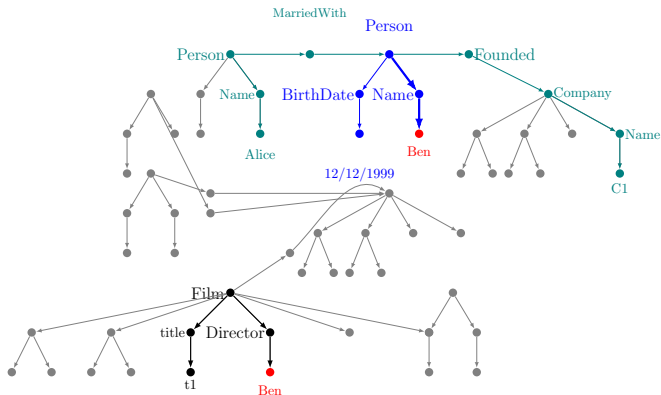
Assembling an InfoCard

1. Find the key context.
2. Retrieve all the information from the key AE
3. Find all the entities that have a relation to the key entity, and retrieve their key context.



Assembling an InfoCard

1. Find the key context.
2. Retrieve all the information from the key AE
3. Find all the entities that have a relation to the key entity, and retrieve their key context.
4. For non-key entities, find their key context



Ranking Metrics

After gathering all relevant information about an entity e , we rank facts based on their informativeness and representativeness.

- **Commonness:** Prioritizes typical properties frequent in similar entities (e.g., "birth date" for a Person).
- **Drift:** Penalizes properties with unusual frequency compared to similar entities (e.g., a person directing significantly more movies than the average director).
- **Informativeness:** Boosts unique values that distinctly identify the entity (e.g., the rarity of a location used as a film production site).

Implementation

- Implemented in Java with PostgreSQL for data storage.
- On the first run, relational tables and key context penalties are computed and cached for faster access.
- Each InfoCard is available as:
 - ▶ CSV file with structured key-value pairs for easy data analysis.
 - ▶ Graphical view for intuitive visual insights.

Results: Quality Analysis

Mondial Dataset: A comprehensive geographic database containing data on **194 countries, 4,139 cities**, rivers, mountains, and organizations.

XML-Specific Challenges:

- Deeply nested structures
- Mixed data types (numeric, string, dates)
- Inconsistent or missing references
- Cyclic and polymorphic references

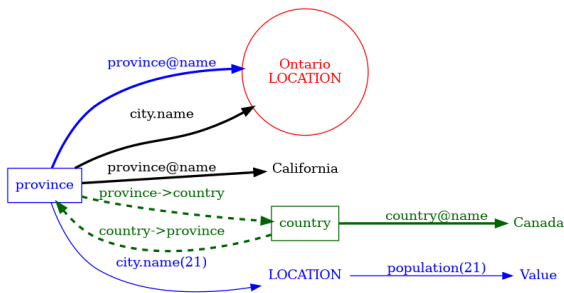


Figure: Ontario's InfoCard

Results: Scalability Analysis

- We evaluate scalability on two datasets: **Movies** and **XMark**, at increasing dataset sizes.
- The execution time for precomputing the core tables (V , O , and R) grows approximately linearly with data size.

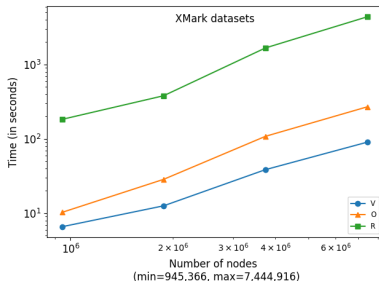
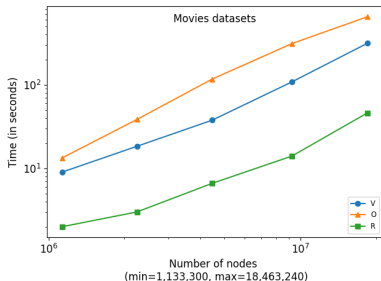




Figure: Scalability of precomputation for Movies and XMark datasets. Each value is averaged over three runs.

Thank you
Questions?

References

-  A. C. Anadiotis et al., Graph Integration of Structured, Semistructured, and Unstructured Data for Data Journalism, in *Information Systems*, vol.104, 2022, 101846.
-  N. Barret, I. Manolescu, and P. Upadhyay, Computing Generic Abstractions from Application Datasets, in *OpenProceedings*, 2024, pp. 94–107.