Data and AI for Investigative Journalism

Ioana Manolescu

CEDAR team, Inria and Institut Polytechnique de Paris, France
ioana.manolescu@inria.fr, @ioanamanol

JDSE 2024

Plan

1. Motivation: Data management for investigative journalism
2. ConnectionLens: Graph-based integration of heterogeneous data
   - So that journalists do not have to know about data models
3. Abstra: Finding the Entity-Relationship conceptual model in a dataset of any model
   - Also a tool for transforming any of these into Property Graphs!
4. Ongoing work and perspectives

Plan

1. Motivation: Data management for investigative journalism
2. ConnectionLens: Graph-based integration of heterogeneous data
   - So that journalists do not have to know about data models
3. Abstra: Finding the Entity-Relationship conceptual model in a dataset of any model
   - Also a tool for transforming any of these into Property Graphs!
4. Ongoing work and perspectives

Not in this talk: statistical fact checking with RadioFrance [5, 6]
(see: `https://team.inria.fr/cedar/projects/statcheck/`)

Part I

Motivation: Investigative Journalism

Why journalism?

- I was born in communist Romania, then a dictatorship.
  Estimated victims 1945-1989: 500.000 to 2M

Why journalism?

- I was born in communist Romania, then a dictatorship.
  Estimated victims 1945-1989: 500.000 to 2M
- Fall of communism in Romania: 1000+ dead, 3000+ wounded.
  No one has been tried / convicted for that.

Why journalism?

- I was born in communist Romania, then a dictatorship.
  Estimated victims 1945-1989: 500.000 to 2M
- Fall of communism in Romania: 1000+ dead, 3000+ wounded.
  No one has been tried / convicted for that.

Functional free press is an important ingredient for **democracy**

- To debate and express dissent
- To analyze and expose society's functioning

Research projects with Le Monde, **radiofrance** since 2015

Journalists vs. the data

- As the world goes digital, journalists stand to gain enormously from **leveraging digital data**

Journalists vs. the data

- As the world goes digital, journalists stand to gain enormously from **leveraging digital data**
- They need to work with the **data they can get their hands on** (Open, or shared by sources, or...)
  - Traditionally, PDFs/HTML that they would read
  - Increasingly, there are also (semi-)structured datasets
  - Heterogeneous data!

Journalists vs. the data

- As the world goes digital, journalists stand to gain enormously from **leveraging digital data**
- They need to work with the **data they can get their hands on** (Open, or shared by sources, or...)
  - Traditionally, PDFs/HTML that they would read
  - Increasingly, there are also (semi-)structured datasets
  - Heterogeneous data!

Paradise Papers:

- Relational database (register of off-shore companies)
- PDFs (contracts stating who represents whom, addresses, lawyers...)
- Emails

Data model heterogeneity

Models and strengths:

- **Tables** CSV, relational (most mature; most regular $\Rightarrow$ optimization)
- **Trees** JSON, XML, K-V: Web content, structured documents, data exchange
- **Graphs** RDF: Open Data; PGs: possibly better suited in business contexts

Data model heterogeneity

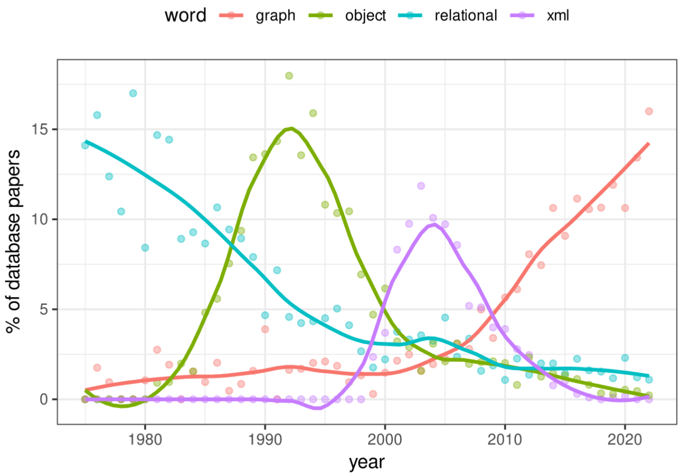
Models and strengths:

- **Tables** CSV, relational (most mature; most regular $\Rightarrow$ optimization)
- **Trees** JSON, XML, K-V: Web content, structured documents, data exchange
- **Graphs** RDF: Open Data; PGs: possibly better suited in business contexts

Support:

	Tables	XML	JSON	KV	RDF	PGs
Relational databases	✓	✓	✓	✓	✓	✓
XML databases		✓				
JSON databases			✓	✓		
KV stores				✓		
RDF databases					✓	✓
PG databases					✓	✓

Data model fashions



Credit: `http://databasearchitects.blogspot.com/2023/02/five-decades-of-database-research.html`

Data heterogeneity: how to live with it?

Data integration: leverage data from several stores (sources)

- Sources may have different data models, schemas, ontologies, query processing capabilities
- Sources may reside on different sites, or one source on many sites

Data integration approaches

- **Mediator**: global schema, logically connected to the local schemas (possibly w/ ontologies). Sources may remain distributed.
- **Warehousing**: all data in a single site, single schema (typically, relational)
- **Data lake**: all data in a single site, no single schema

Data heterogeneity: how to live with it?

Data integration: leverage data from several stores (sources)

- Sources may have different data models, schemas, ontologies, query processing capabilities
- Sources may reside on different sites, or one source on many sites

Data integration approaches

- **Mediator**: global schema, logically connected to the local schemas (possibly w/ ontologies). Sources may remain distributed.
- **Warehousing**: all data in a single site, single schema (typically, relational)
- **Data lake**: all data in a single site, no single schema

Data heterogeneity: how to live with it?

Data integration: leverage data from several stores (sources)

- Sources may have different data models, schemas, ontologies, query processing capabilities
- Sources may reside on different sites, or one source on many sites

Data integration approaches

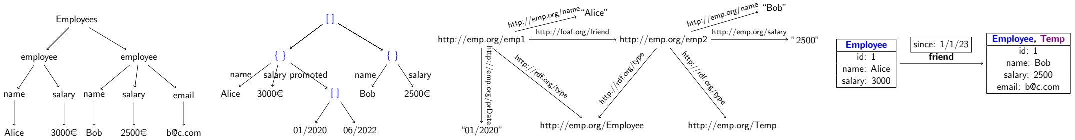
- **Mediator**: global schema, logically connected to the local schemas (possibly w/ ontologies). Sources may remain distributed.
- **Warehousing**: all data in a single site, single schema (typically, relational)
- **Data lake**: all data in a single site, no single schema

This talk

1. Lake-style integration under graph data model
2. Finding interesting connections in such graphs
3. Making sense of such graphs as PGs

Making sense of heterogeneous data for journalism

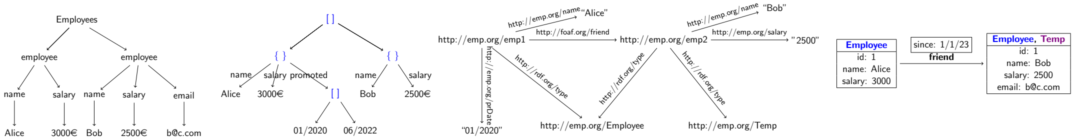
- Journalists' application domain follows the topic of interest: news cycle, or investigation topic.
- Journalists use whatever data they can get their hands on. Different data models (+ text, Office, etc.)



- Journalists are familiar with text and documents > spreadsheet ≫ anything else
- Data producers often uncollaborative ⇒ documentation, schema missing

Making sense of heterogeneous data for journalism

- Journalists' application domain follows the topic of interest: news cycle, or investigation topic.
- Journalists use whatever data they can get their hands on. Different data models (+ text, Office, etc.)



- Journalists are familiar with text and documents $>$ spreadsheet $\gg$ anything else
- Data producers often uncollaborative $\Rightarrow$ documentation, schema missing
- Data understanding conditions even the earliest stages of journalistic work!

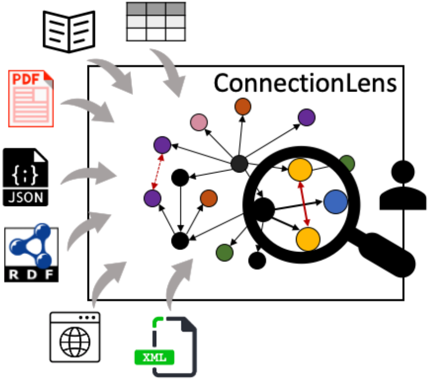
Part II

ConnectionLens: Graph-Based Heterogeneous Data Integration

Joint work with O. Balalau, A. Anadiotis, M. Mohanty (Inria), H. Galhardas (INESC), and many others

ConnectionLens: integrating data into graphs [2]

1. Focus on (semi)structured data formats: RDBs, CSV, JSON, XML, RDF, PGs, ...
2. Enrich the graph with Named Entities extracted from each text (value) node.



Papers and code: `https://team.inria.fr/cedar/connectionlens/`

Relational data conversion to a graph

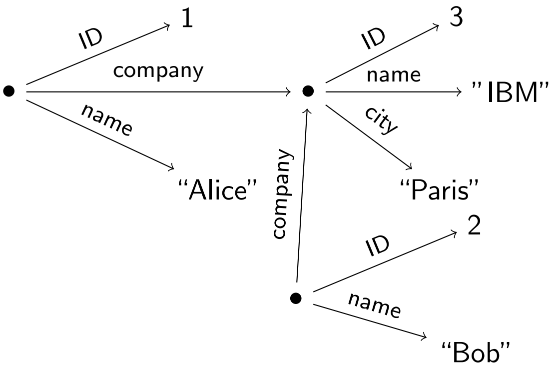
Relational model (also **CSV**): tables

Employees

ID	name	company
1	Alice	3
2	Bob	3

Companies

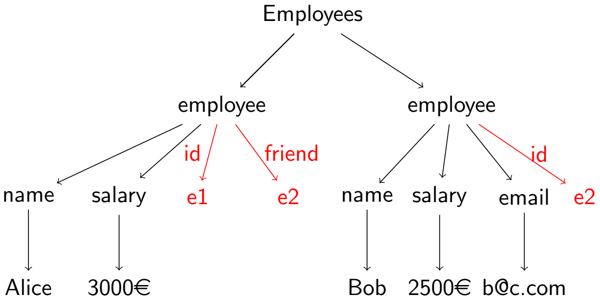
ID	name	city
3	IBM	Paris



XML documents: mostly trees

XML elements may have attributes

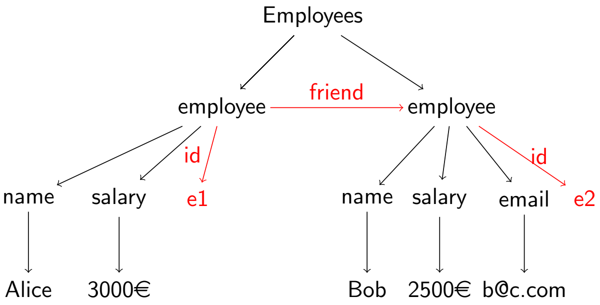
```
<Employees>
    <employee id="e1" friend="e2">
        <name>Alice</name>
        <salary>3000€</salary>
    </employee>
    <employee id="e2">
        <name>Bob</name>
        <salary>2500€</salary>
        <email>b@c.com</email>
    </employee>
</Employees>
```



XML documents: mostly trees

A schema (Document Type Description or XML Schema) may state that id is a PK, and friend is a FK

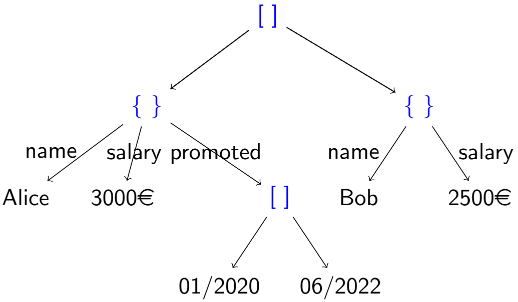
```
<Employees>
  <employee id="e1" friend="e2">
    <name>Alice</name>
    <salary>3000€</salary>
  </employee>
  <employee id="e2">
    <name>Bob</name>
    <salary>2500€</salary>
    <email>b@c.com</email>
  </employee>
</Employees>
```



JSON documents: trees

Main constructs: maps (objects) [ ] and arrays (lists) { }

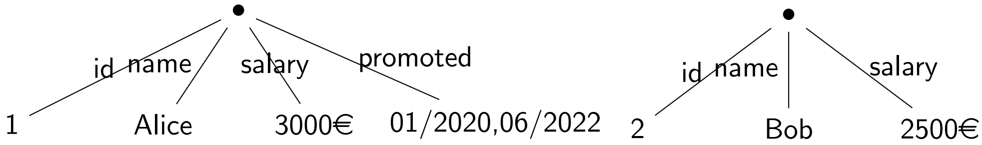
```
[
  {"id": "1", "name": "Alice",
   "salary": "3000",
   "promoted": ["01/2020", "06/2022"]},
 {"id": "2", "name": "Bob",
   "salary": "2500"},
]
```



Java**S**cript **O**bject **N**otation

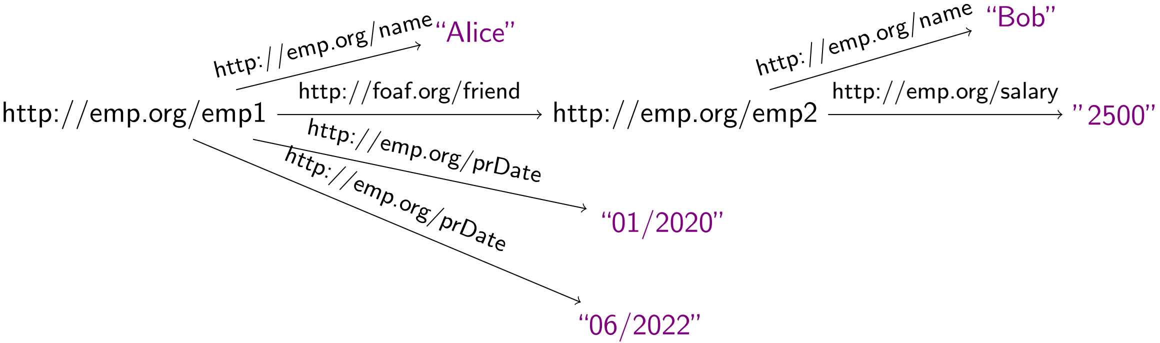
Key-value pairs are simplified JSON (maps only)

Among the simplest imaginable data models.



RDF graphs

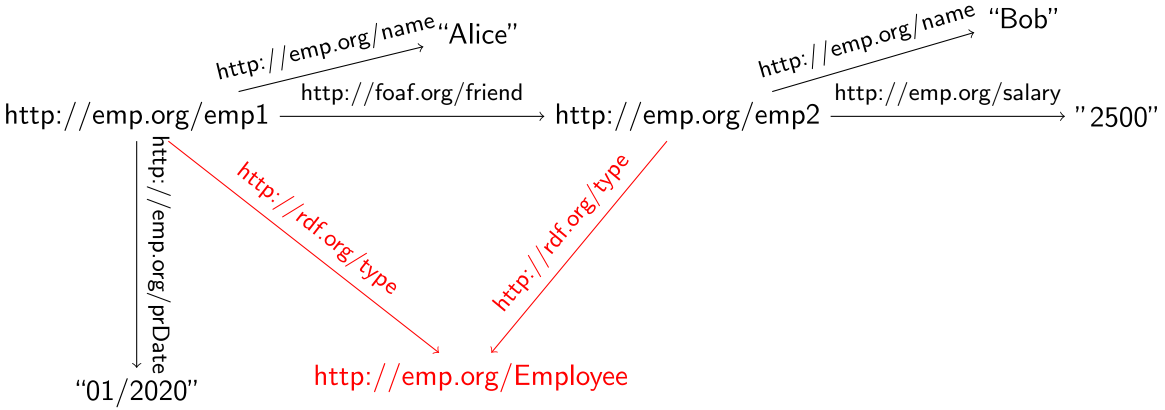
RDF: **R**esource **D**escription **F**ramework, resources identified by International Resource Identifiers (IRIs)
Resources described by properties (IRIs) w/ atomic values. A value can be an IRI or a literal.



Adding meaning to RDF graph: types

They are optional

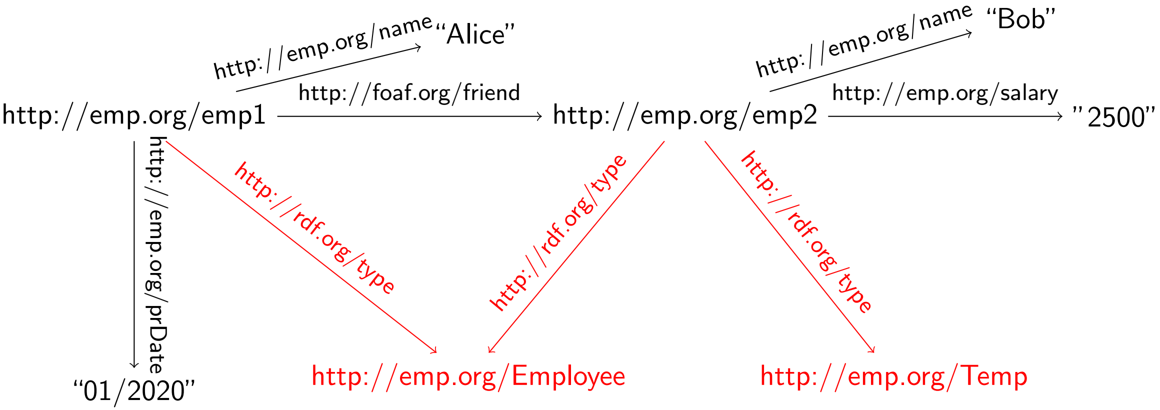
A resource can have zero or more types which can be related or not



Adding meaning to RDF graphs: types

They are optional

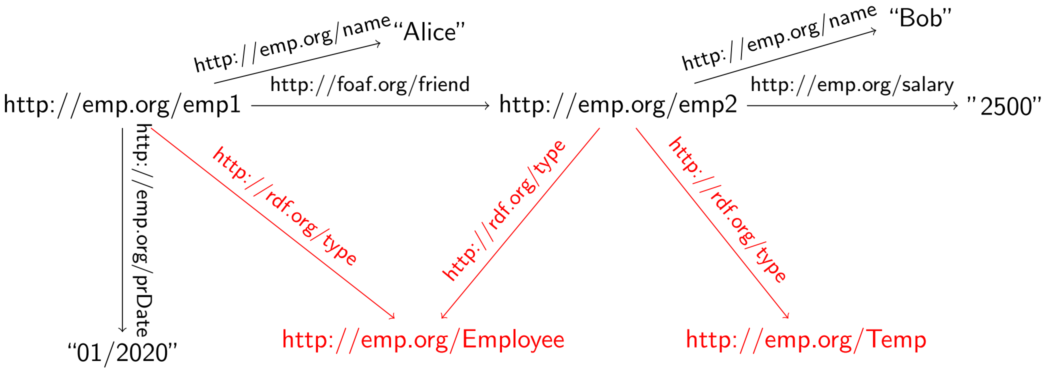
A resource can have zero or more types which can be related or not



Adding meaning to RDF graphs: ontologies

Optional, allow describing how types and properties are related. Simplest examples:

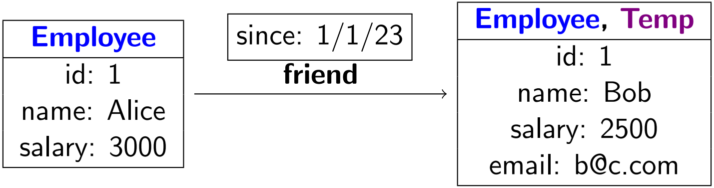
- http://emp.org/Temp is a subclass of http://emp.org/Employee'
- any resource having a http://emp.org/salary is an http://emp.org/Employee



Property graphs

Unlike RDF nodes, PG nodes contain their attributes.
Nodes can have zero or more labels. Relationships can also have attributes.

Industry first (Neo4J), then research, then standards [1, 16]



Entity extraction [2]

Extract Named Entities (NEs) from any text field originating from any data model

- **Pattern-based** for: emails, URIs, hashtags, ...
- Using **language models** for People, Locations, Organizations
  - Used Flask EN model, trained a Flask FR model [2]
  - Prompted ChatGPT4 [11], Gemma2 [17] for extraction

Entity extraction [2]

Extract Named Entities (NEs) from any text field originating from any data model

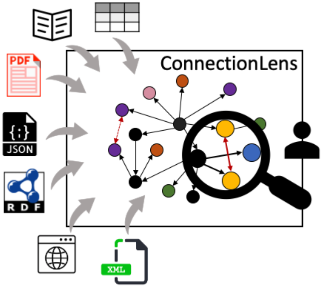
- **Pattern-based** for: emails, URIs, hashtags, ...
- Using **language models** for People, Locations, Organizations
  - Used Flask EN model, trained a Flask FR model [2]
  - Prompted ChatGPT4 [11], Gemma2 [17] for extraction

Disambiguate NEs in their context against a KG

Entity extraction [2]

Extract Named Entities (NEs) from any text field originating from any data model

- **Pattern-based** for: emails, URIs, hashtags, ...
- Using **language models** for People, Locations, Organizations
  - Used Flask EN model, trained a Flask FR model [2]
  - Prompted ChatGPT4 [11], Gemma2 [17] for extraction

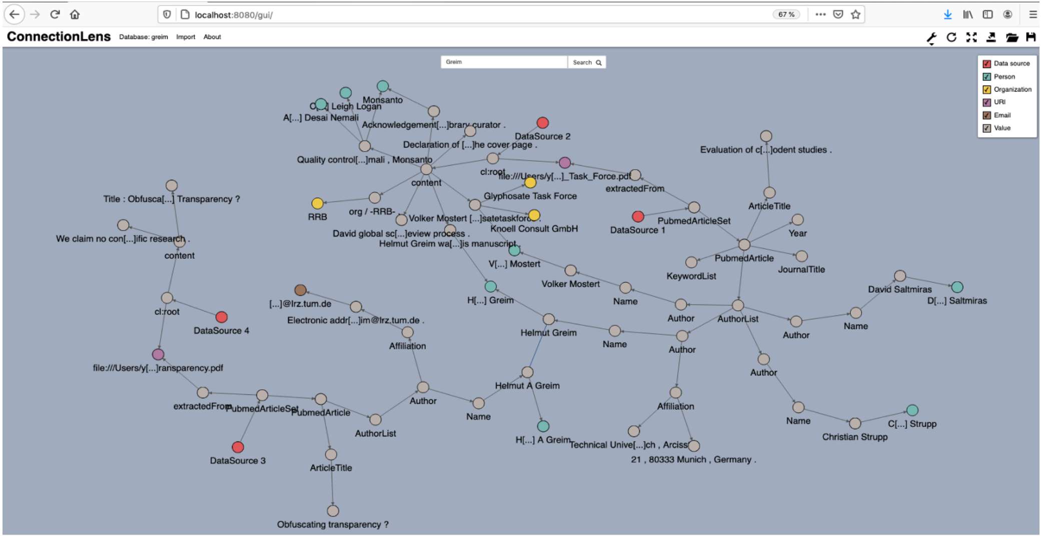


Disambiguate NEs in their context against a KG

For each distinct entity, we add to the graph one entity node, with an incoming extraction edge from each node in which it has been found

Entities may lead to connections across datasets! (and more connections within)

Sample graph: conflicts of interest in the biomedical domain



Sample graph: conflicts of interest in the biomedical domain [2]

A graph for studying conflicts of interest in the biomedical domain (thanks S. Horel, Le Monde)

- PubMed publication notices (XML): authors, employers, Conflict of Interest statements (People, Organization)
- Articles in PDF → JSON: Acknowledgments or Disclosure paragraphs (People, Organization)
- HTML: a Web crawl of media articles on health topics

$\|N\|$	$\|E\|$	$\|N\|$	$\|N_P\|$	$\|N_O\|$	$\|N_L\|$
XML	32,028,429	19,851,904	1,483,631	584,734	126,629
JSON	1,025,307	432,303	75,297	7,320	4,139
HTML	246,636	185,479	3,726	7,227	320
Total	33,300,372	20,469,686	1,562,654	665,167	131,088

Table 3: Statistics on Conflict of Interest application graph.

Building ConnectionLens graphs [2]

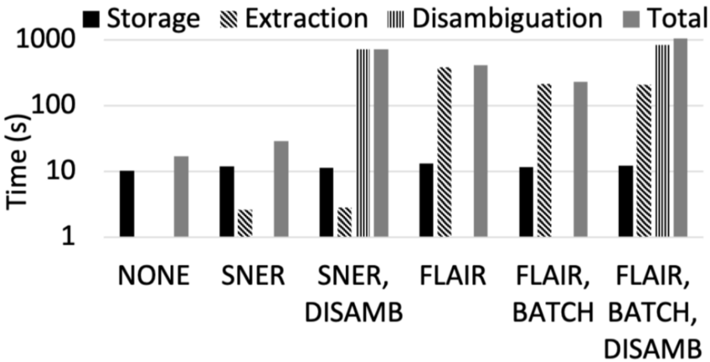


Figure 6: Graph construction time (seconds).

Building ConnectionLens graphs [2]

NE extraction time dominates

Reduced via: caching, parallelization (batching), GPU, learned when not to extract [14]

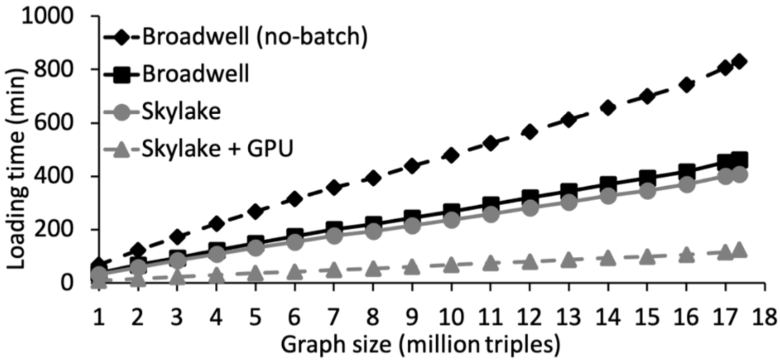


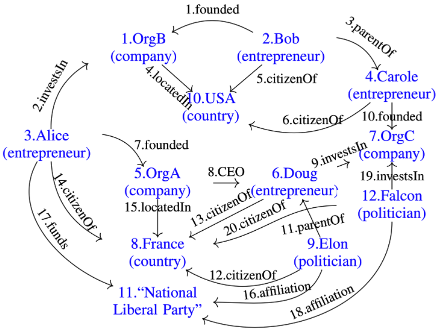
Figure 7: YAGO loading time (minutes) using Flair.

Keyword search in ConnectionLens graphs

Keyword search: given k keywords, find minimal connecting trees, each tree containing one node matching each keyword
NP-hard problem (Group Steiner Trees)

Efficient **keyword search algorithms** [4, 3]

- **Heuristic**, complete for $k \leq 3$ and a large family of solutions for higher k [4], works with any connection score function
- Multi-threaded algorithm [3]

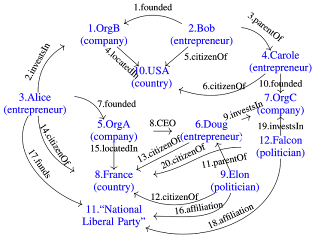


Integrating keyword search into a graph query language [4]

- SPARQL allows verifying the presence of paths (regex) between two variables
- GQL allows verifying the presence of arbitrary paths, with many flexible options
- We add the ability to (1) return (2) paths or trees.

How are the US entrepreneurs, French entrepreneurs and French politicians related?

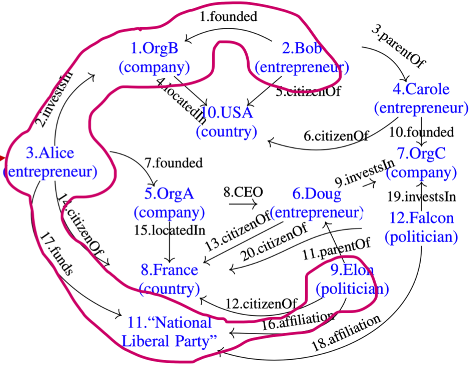
```
MATCH
(x WHERE x.type=entrepreneur)-[a:citizenOf]->(b:  USA),
(y WHERE y.type=entrepreneur)-[c:citizenOf]->(d:  France),
(z WHERE z.type=politician)-[e:citizenOf]->(f:  France),
(x, y, z, w)
RETURN w
```



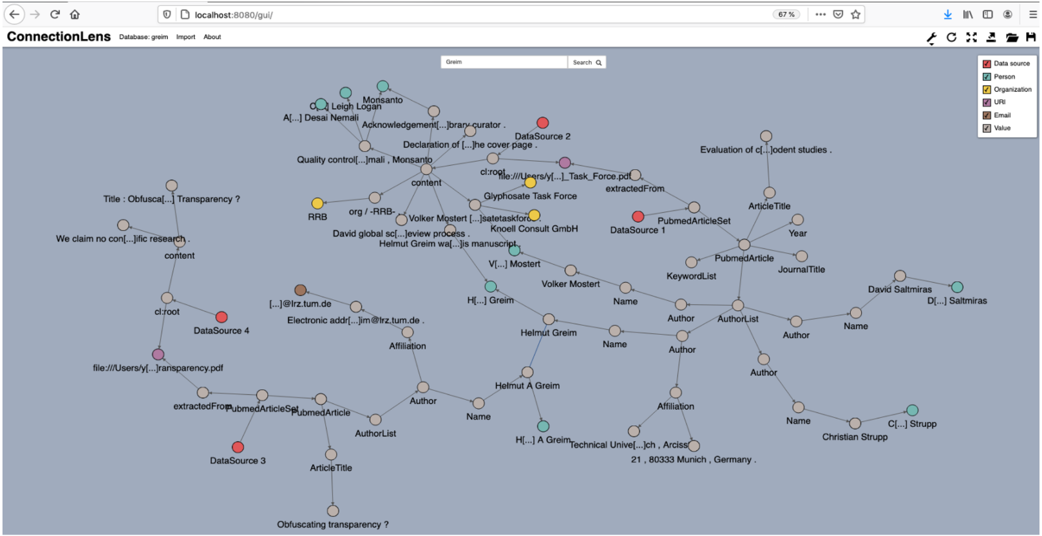
Integrating keyword search into a graph query language [4]

```
MATCH
(x WHERE x.type=entrepreneur)-[a:citizenOf]->(b: USA),
(y WHERE y.type=entrepreneur)-[c:citizenOf]->(d: France),
(z WHERE z.type=politician)-[e:citizenOf]->(f: France), (x, y, z, w)
RETURN w
```

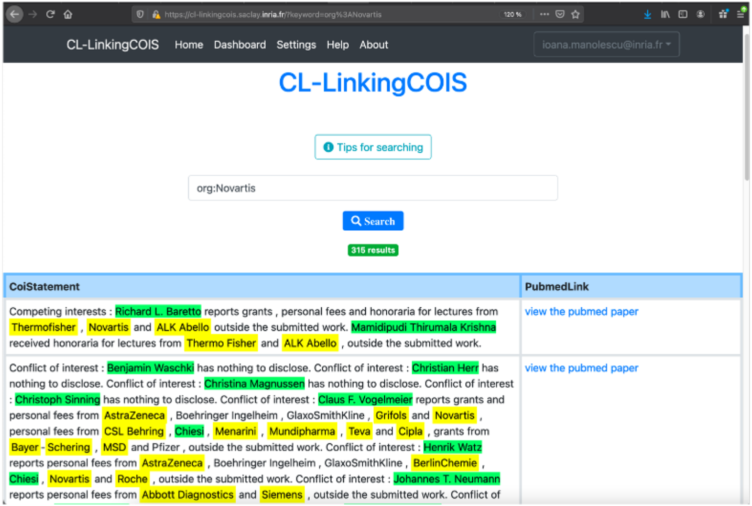
x	y	z	w
Bob	Alice	Elon	
Carole	Doug	Falcon	
Carole	Alice	Falcon	....
...	...	...	...



Unfortunately, journalists don't like such views of the data!

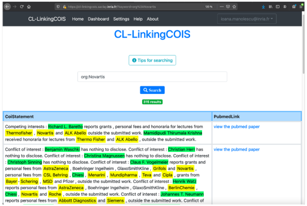


Journalists prefer simpler views over ConnectionLens graphs



Towards data abstraction

- Journalists (and other technical users) need help figuring out what a dataset contains! ABSTRA (see next)
- Based on this knowledge, their information needs can be met through:
  - Automatically identifying interesting entity paths: PATHWAYS [9, 10]
  - Abstraction-based GUI that enables formulating queries: CONNECTIONSTUDIO [7]



Part III

ABSTRA: Abstracting Data of Any Model

Joint work with Nelly Barret (Inria), Prajna Upadhyay (Inria, now IIT Khanpur)

How to help journalists understand heterogeneous data

Focus on (semi)structured data formats: RDBs, CSV, JSON, XML, RDF, PGs, ...

Core principles

1. Do not expose data model details.

How to help journalists understand heterogeneous data

Focus on (semi)structured data formats: RDBs, CSV, JSON, XML, RDF, PGs, ...

Core principles

1. Do not expose data model details.
2. Any application dataset (tabular/RDB, tree- or graph-oriented) contains some entities connected by some relationships. Find and (graphically) show these!

How to help journalists understand heterogeneous data

Focus on (semi)structured data formats: RDBs, CSV, JSON, XML, RDF, PGs, ...

Core principles

1. Do not expose data model details.
2. Any application dataset (tabular/RDB, tree- or graph-oriented) contains some entities connected by some relationships. Find and (graphically) show these!
    - Entities may have nested structure

How to help journalists understand heterogeneous data

Focus on (semi)structured data formats: RDBs, CSV, JSON, XML, RDF, PGs, ...

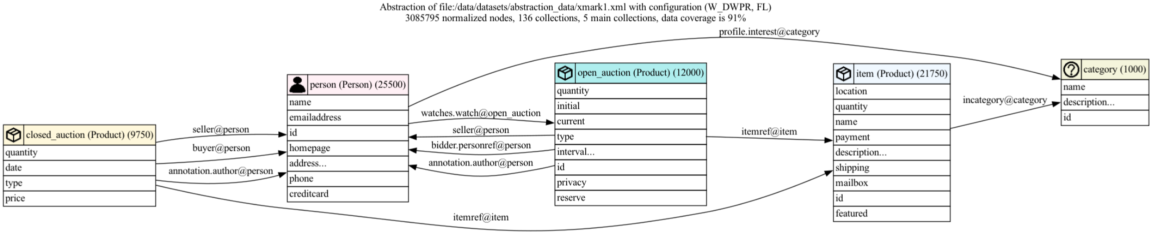
Core principles

1. Do not expose data model details.
2. Any application dataset (tabular/RDB, tree- or graph-oriented) contains some entities connected by some relationships. Find and (graphically) show these!
   - Entities may have nested structure
3. Let users control how much information they want.

ABSTRA project [10, 12, 13]. Code and (many) examples:

`https://team.inria.fr/cedar/projects/abstra/`

Sample abstraction of an XMark XML document



Application benchmark: online auctions site

3M nodes, 80 different labels, on 124 labeled paths

Dataset abstraction stages

1. Turn any dataset into a ConnectionLens graph
2. Normalize the graph: all edges become unlabeled
3. Summarize the graph: equivalent nodes grouped in collections
4. Identify (nested) entities:
   - Select some collections as entity roots
   - Determine each entity's boundaries
5. Classify entities: assign a human-understandable name
6. Identify relationships between entities

Data graph summarization: which groups of nodes are equivalent?

Reverse each data model's guidelines for representing "similar things":

Relations all nodes representing tuples of the same table

Data graph summarization: which groups of nodes are equivalent?

Reverse each data model's guidelines for representing "similar things":

- **Relations** all nodes representing tuples of the same table
- **CSV** all nodes representing rows

Data graph summarization: which groups of nodes are equivalent?

Reverse each data model's guidelines for representing "similar things":

- **Relations** all nodes representing tuples of the same table
- **CSV** all nodes representing rows
- **XML** all XML elements w/ same name

Data graph summarization: which groups of nodes are equivalent?

Reverse each data model's guidelines for representing "similar things":

- **Relations** all nodes representing tuples of the same table
- **CSV** all nodes representing rows
- **XML** all XML elements w/ same name
- **JSON** all nodes on the same path from the root

Data graph summarization: which groups of nodes are equivalent?

Reverse each data model's guidelines for representing "similar things":

- **Relations** all nodes representing tuples of the same table
- **CSV** all nodes representing rows
- **XML** all XML elements w/ same name
- **JSON** all nodes on the same path from the root
- **RDF** (*i*) typed nodes are equivalent if they have the same set of types; (*ii*) untyped nodes are grouped according to a flexible notion of property cliques [18]

Data graph summarization: which groups of nodes are equivalent?

Reverse each data model's guidelines for representing "similar things":

- **Relations** all nodes representing tuples of the same table
- **CSV** all nodes representing rows
- **XML** all XML elements w/ same name
- **JSON** all nodes on the same path from the root
- **RDF** (*i*) typed nodes are equivalent if they have the same set of types; (*ii*) untyped nodes are grouped according to a flexible notion of property cliques [18]
- **PGs** (*i*) nodes with the same label set are equivalent; (*ii*) relationship nodes corresponding to relationships of the same type are equivalent

Data graph summarization: which groups of nodes are equivalent?

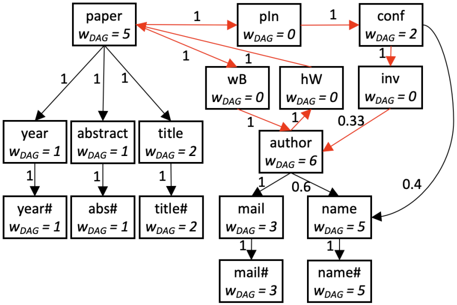
Reverse each data model's guidelines for representing "similar things":

- **Relations** all nodes representing tuples of the same table
- **CSV** all nodes representing rows
- **XML** all XML elements w/ same name
- **JSON** all nodes on the same path from the root
- **RDF** (*i*) typed nodes are equivalent if they have the same set of types; (*ii*) untyped nodes are grouped according to a flexible notion of property cliques [18]
- **PGs** (*i*) nodes with the same label set are equivalent; (*ii*) relationship nodes corresponding to relationships of the same type are equivalent

And: for all $x_1 \equiv x_2$ and label a, if $x_1 \xrightarrow{a} y_1, x_2 \xrightarrow{a} y_2$ and y_1, y_2 are leaves, then $y_1 \equiv y_2$.

Graph summarization result: collection graph

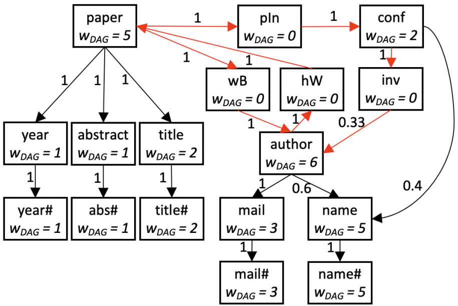
Each set of equivalent nodes is a collection; collections make up the collections graph



Red edges belong to cycles

year# is the collection of values of the nodes in the year collection

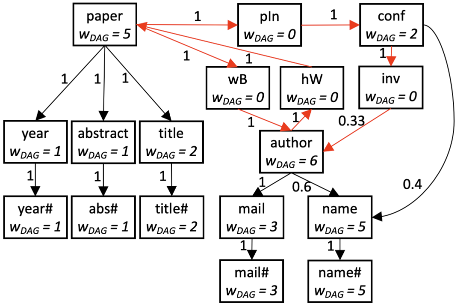
Selecting entities and their boundaries



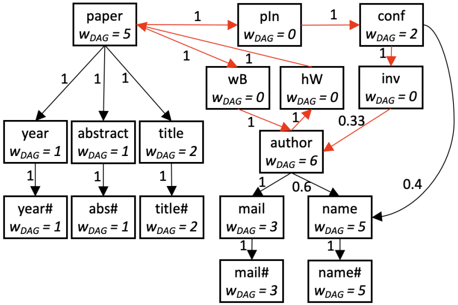
We need to:

- Select some collections as entity roots;
- For each selected entity root, determine which other collections are in its boundary
- Until we identify k entities and/or $f\%$ of the data is reflected in the returned entities.

Which collections make good entity roots?

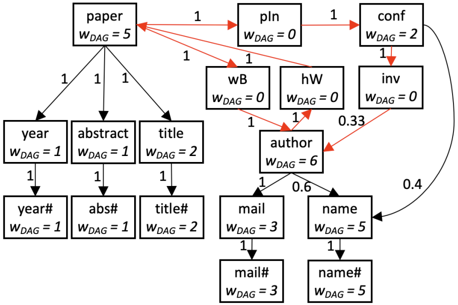


Which collections make good entity roots?



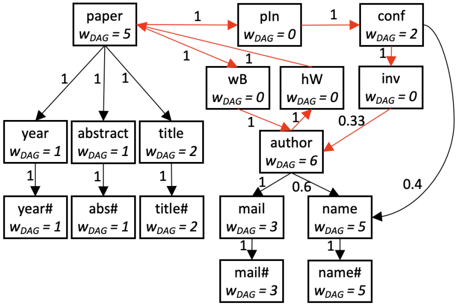
- Must contain more than one node

Which collections make good entity roots?



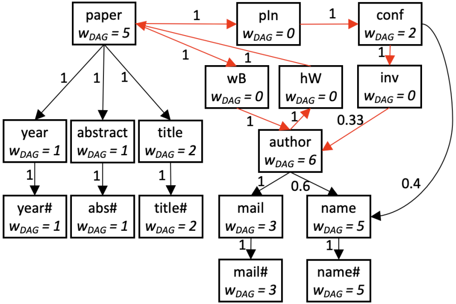
- Must contain more than one node
- The more attributes, the better

Which collections make good entity roots?



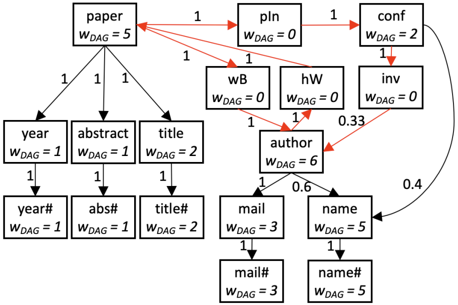
- Must contain more than one node
- The more attributes, the better
- The more descendant attributes, the better
  $\Rightarrow$ backward data weight propagation

Which collections make good entity roots?



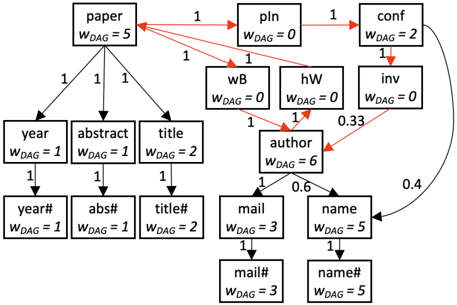
- Must contain more than one node
- The more attributes, the better
- The more descendant attributes, the better
  $\Rightarrow$ backward data weight propagation
- More general: compute **PageRank** on the inverse collection graph: nodes with many descendant attributes are favored

Which collections make good entity roots?



- Must contain more than one node
- The more attributes, the better
- The more descendant attributes, the better
  $\Rightarrow$ backward data weight propagation
- More general: compute **PageRank** on the inverse collection graph: nodes with many descendant attributes are favored
- Pick the highest-rank collection as the root of the next selected entity

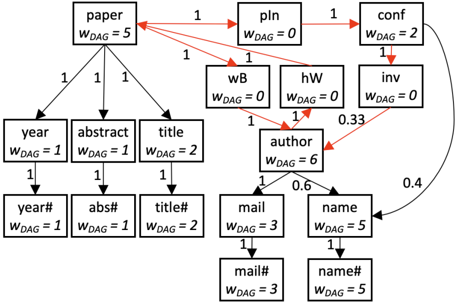
Which collections make good entity roots?



- Must contain more than one node
- The more attributes, the better
- The more descendant attributes, the better
  $\Rightarrow$ backward data weight propagation
- More general: compute **PageRank** on the inverse collection graph: nodes with many descendant attributes are favored
- Pick the highest-rank collection as the root of the next selected entity

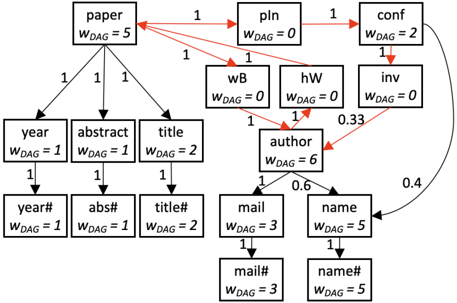
Assume **paper** is selected.

What to include in an entity boundary?



In the boundary of the entity rooted in c, we include any other collection c'

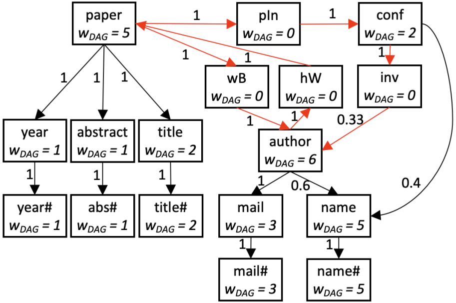
What to include in an entity boundary?



In the boundary of the entity rooted in c, we include any other collection c'

- Such that there is a path $c \rightsquigarrow c'$ with no in-cycle edges

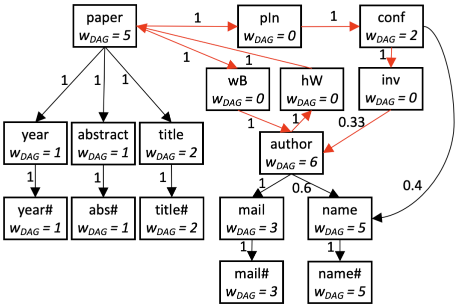
What to include in an entity boundary?



In the boundary of the entity rooted in c, we include any other collection c'

- Such that there is a path $c \rightsquigarrow c'$ with no in-cycle edges
- And along this path, each edge $c_i \rightarrow c_j$ satisfies:
  - Each node in c_i has at most one child in c_j; and/or
  - At least $x\%$ of the nodes in c_j have a parent in c_i.

What to include in an entity boundary?

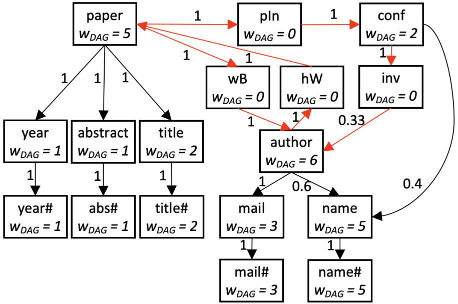


In the boundary of the entity rooted in c, we include any other collection c'

- Such that there is a path $c \rightsquigarrow c'$ with no in-cycle edges
- And along this path, each edge $c_i \rightarrow c_j$ satisfies:
  - Each node in c_i has at most one child in c_j; and/or
  - At least $x\%$ of the nodes in c_j have a parent in c_i.

The **paper** entity includes: year, abstract, title (allowing in-cycle edges: all the collections)

Selecting the entities

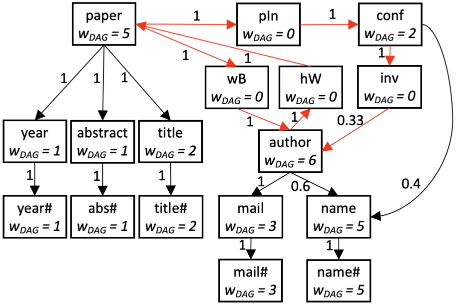


<u>repeat</u>

- Compute PageRank on collection graph
- Highest-rank node $\rightarrow$ root of next entity e
- Assign other collections in e's boundary
- Update the collection graph:
  - remove e's root + outgoing edges
  - remove every collection in e's boundary that contributed only to it
  - reduce the weights of the other collections in e's boundary

<u>until</u> k entities selected and/or $f\%$ of the data is reflected

Finding the relationships



- <u>For each</u> c_1, root of entity e_1
  - <u>For each</u> path $c_1 \rightsquigarrow c_2$ where c_2 is the root of e_2
  - Create a relationship $e_1 \rightarrow e_2$ with the labels on the path

Classifying entities

Goal: compute for each entity a human-understandable label

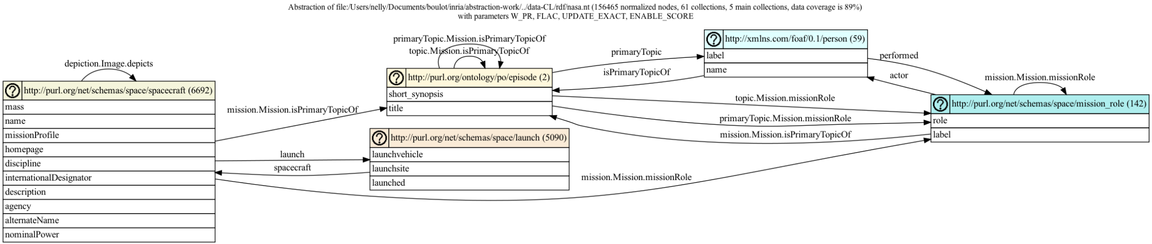
Ingredients

- Vocabulary of commonly-used classes and properties (from open Knowledge Graphs)
- GitTables: table and column names from GitHub

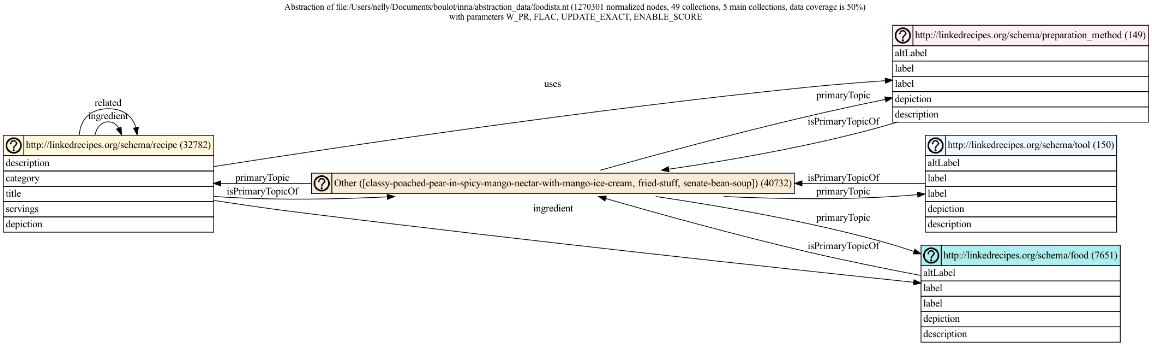
Method

1. Identify, for People, Location, Organization, strongly related KG classes (manual)
2. Leverage NEs (P/L/O) extracted from `name`, `description` etc. attributes to propose type(s) (α) for the parent of such attribute.
3. Match the names of entity root nodes with class and table names (Word2Vec) (β)
4. Match the names of attributes of entity root nodes, with names of properties, and of GitTables columns (γ)
5. Assign a type through majority vote (α, β, γ)

Abstraction of NASA RDF graph



Abstraction of Foodista RDF graph



Part IV

Conclusion and perspectives

Wrap-up

Summary

- Data heterogeneity remains a reality. Particular data models are preferrable for some applications.
- Heterogeneity is a big (and, I claim, useless) hurdle for non-expert users.
- ConnectionLens: integrate data of heterogeneous models into a graph, densified by NE connections; keyword search; automatic path recommendation [9, 11]
- Abstra: Find entities and relationships in application datasets regardless of original data model [13]

Ongoing/perspectives

- Use Abstra to transform any dataset into a Property Graph [8]
- Leverage this for interactive data exploration (w/ T. Bouganim and E. Pietriga) [15]
- Exploration of very large corpora of heterogeneous datasets

References I

[1] GQL (graph query language) standard, ISO/IEC 39075:2024.
https://www.iso.org/standard/76120.html, 2024.

[2] Angelos Anadiotis, Oana Balalau, Catarina Conceicao, et al.
Graph integration of structured, semistructured and unstructured data for data journalism.
Information Systems, 104, 2022.

[3] Angelos-Christos Anadiotis, Oana Balalau, Théo Bouganim, Francesco Chimienti, Helena Galhardas, Mhd Yamen Haddad, Stéphane Horel, Ioana Manolescu, and Youssr Youssef.
Empowering Investigative Journalism with Graph-based Heterogeneous Data Management.
Bulletin of the Technical Committee on Data Engineering, September 2021.

[4] Angelos Christos Anadiotis, Ioana Manolescu, and Madhulika Mohanty.
Integrating Connection Search in Graph Queries.
In ICDE 2023 - 39th IEEE International Conference on Data Engineering, Anaheim (CA), United States, April 2023.

[5] Oana Balalau, Simon Ebel, Helena Galhardas, Théo Galizzi, and Ioana Manolescu.
STaR: Space and Time-aware Statistic Query Answering.
In CIKM 2024 - 33rd ACM International Conference on Information and Knowledge Management, Boise, Idaho, United States, October 2024.

[6] Oana Balalau, Simon Ebel, Théo Galizzi, Ioana Manolescu, Quentin Massonnat, Antoine Deiana, Emilie Gautreau, Antoine Krempf, Thomas Pontillon, Gérald Roux, and Joanna Yakin.
Statistical Claim Checking: StatCheck in Action (demonstration).
In CIKM 2022 - 31st ACM International Conference on Information and Knowledge Management, Atlanta / Hybrid, United States, October 2022.

[7] Nelly Barret, Simon Ebel, Théo Galizzi, Ioana Manolescu, and Madhulika Mohanty.
User-friendly exploration of highly heterogeneous data lakes.
In CoopIS 2023 - International Conference on Cooperative Information Systems, Groningen, Netherlands, October 2023.

References II

[8] Nelly Barret, Tudor Enache, Ioana Manolescu, and Madhulika Mohanty.
Finding the PG schema of any (semi)structured dataset: a tale of graphs and abstraction.
In
SEAGraph Workshop 2024 - 3rd Workshop on Search, Exploration, and Analysis in Heterogeneous Datastores - 40th IEEE International Conference on Data Eng
Utrecht, Netherlands, May 2024.

[9] Nelly Barret, Antoine Gauquier, Jia Jean Law, and Ioana Manolescu.
Exploring heterogeneous data graphs through their entity paths.
In ADBIS, 2023.

[10] Nelly Barret, Antoine Gauquier, Jia Jean Law, and Ioana Manolescu.
Pathways: entity-focused exploration of heterogeneous data graphs (demonstration).
In ESWC, 2023.

[11] Nelly Barret, Antoine Gauquier, Jia Jean Law, and Ioana Manolescu.
Finding meaningful paths in heterogeneous graphs with pathways.
Information Systems, 2024.

[12] Nelly Barret, Ioana Manolescu, and Prajna Upadhyay.
Abstra: toward generic abstractions for data of any model (demonstration).
In CIKM, 2022.

[13] Nelly Barret, Ioana Manolescu, and Prajna Upadhyay.
Computing generic abstractions from application datasets.
In EDBT, 2024.

References III

[14] Théo Bouganim, Ioana Manolescu, and Helena Galhardas.
Efficiently Identifying Disguised Missing Values in Heterogeneous, Text-Rich Data.
In Transactions on Large-Scale Data- and Knowledge-Centered Systems LI, volume 13410 of Lecture Notes in Computer Science, pages 97–118. Springer Berlin Heidelberg, October 2022.

[15] Théo Bouganim, Ioana Manolescu, and Emmanuel Pietriga.
A Unified Visual Exploration Framework for (Semi-)structured Data.
In BigVis 2024 - 7th International Workshop on Big Data Visual Exploration and Analytics (VLDB 2024), Guangzhou, China, August 2024. VLDB.

[16] Alin Deutsch, Nadime Francis, Alastair Green, Keith Hare, Bei Li, Leonid Libkin, et al.
Graph pattern matching in GQL and SQL/PGQ.
In SIGMOD, 2022.

[17] Minh Hoang Duong and Ioana Manolescu.
Named entity extraction in connectionlens based on large language models.
Junior Data Science and Engineering Conference, 2024.

[18] François Goasdoué, Pawel Guzewicz, and Ioana Manolescu.
RDF graph summarization for first-sight structure discovery.
The VLDB Journal, 29(5), April 2020.