PathWays: entity-focused exploration of heterogeneous data graphs

Nelly Barret[1][0000−0002−3469−4149], Antoine Gauquier[2], Jia-Jean Law[3], and Ioana Manolescu[1][0000−0002−0425−2462]

[1] Inria and Institut Polytechnique de Paris, France
nelly.barret@inria.fr, ioana.manolescu@inria.fr
[2] Institut Mines Télécom, France antoine.gauquier@etu.imt-nord-europe.fr
[3] Ecole Polytechnique, France jia-jean.law@polytechnique.edu

Abstract. Graphs, and notably RDF graphs, are a prominent way of sharing data. As data usage democratizes, users need help figuring out the useful content of a graph dataset. In particular, journalists with whom we collaborate [3] are interested in identifying, in a graph, the *connections between entities*, e.g., people, organizations, emails, etc.
We propose PathWays, an interactive tool for exploring data graphs through *their data paths connecting Named Entities (NEs, in short)*; each data path leads to a tabular-looking set of results. NEs are extracted from the data through dedicated Information Extraction modules. PathWays leverages the ConnectionLens platform [4,6] and follow-up work on dataset abstraction [5]. Its novelty lies in its interactive and efficient approach to enumerate, compute, and analyze NE paths.

Keywords: Data graphs · Graph exploration · Information Extraction

1 Motivation and Problem Statement

Data graphs, including RDF knowledge graphs, as well as Property Graphs (PGs), are often used to represent and share data. More generally, *any structured or semistructured dataset can be viewed as a graph*, having: (*i*) an internal node for each structure element of the original dataset, e.g., relational tuple, XML element or attribute, JSON map or array, URI in an RDF graph; (*ii*) a leaf node for each value in the dataset, e.g., attribute value in a relational tuple, text node or attribute value in XML, atomic (leaf) value in a JSON document, or literal in RDF. The connections between the data items in the original dataset lead to edges in the graph, e.g. parent-child relationship between XML or JSON nodes, edges connecting each relational tuple node with their attributes, etc. In a relational database, when primary key-foreign key pairs are present, they lead to further edges allowing one tuple, e.g., an Employee, to "point to" the Company tuple representing their employer. This graph view of a dataset has been introduced to support unstructured (keyword-based) search on (semi)structured data, since [2,8] and through many follow-up works [12].

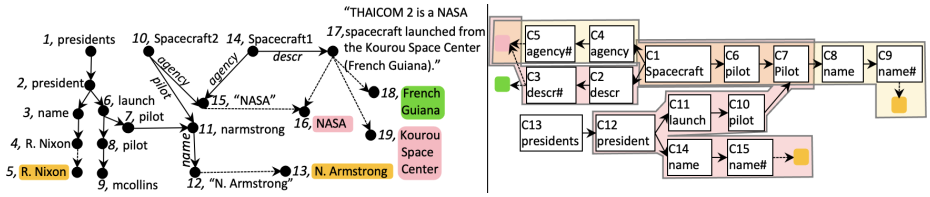

Fig. 1. Sample data graph (left), and corresponding collection graph (right) on which paths linking entities are explored (highlighted areas).

1.1 Entity-rich graphs

Building on this idea of integrating heterogeneous data into graphs, the ConnectionLens system [4,6] has been developed to facilitate, for users lacking IT skills, such as data journalists, the exploration of datasets of various models, including relational/CSV, XML, JSON, RDF, and PGs. ConnectionLens turns *any (set of) datasets into a single graph as outlined above*. For instance, the data graph at left in Fig. 1 features RDF triples about NASA spacecrafts (labeled edges), and an XML document describing presidents who attended spacecraft launches (tree with labeled node and unlabeled edges). ConnectionLens also includes Information Extraction modules, which extract, from any leaf node in the data graph, Named Entities (People, Locations and Organizations) [4], as well as other types of entities that journalists find interesting: temporal moments (date, time); Web site URIs; email addresses; and hashtags. We designate any of these pieces of information as *entities*, and we model them as extra nodes, e.g., in the data graph in Fig. 1, organizations appear on a pink background, people on yellow and locations on green, respectively. Each entity is extracted from a leaf text node, to which it is connected by a dashed edge. When an entity is extracted from more than one text node, the edges connecting it to those nodes increase the connectivity of the graph, e.g., "NASA" extracted from the nodes with IDs 15 and 17.

1.2 Entity paths

Journalists are interested in *the paths connecting entities* in a given dataset. For instance, in Fig. 1, they may want to know "how people are connected to places". Similarly, an article describes French real estate bought by family members of dictators abroad; here, journalists ask "what are the paths between people and cities (where real estate is)?". Importantly, we should consider paths *irrespective of the direction of the edges in the data graph*. This is because, depending on how the data is modeled, we may encounter $x \xrightarrow{\text{boughtProperty}} y \xrightarrow{\text{locatedIn}} c$, or $x \xleftarrow{\text{hasOwner}} y \xrightarrow{\text{locatedIn}} c$; both paths are interesting.

Goal: efficient exploration of entity connections Journalist questions such as those above ask for *data paths ending in entity pairs of certain kinds*. When shown the set of corresponding labeled paths, users may pick one to *further*

explore: how many pairs of entities are connected by each path? which entities are most frequent, e.g., in which cities are most properties located? how do the cities spread across countries, etc. Such analysis requires *materializing the entity pairs connected by the paths*, which may be very costly, if (*i*) the graph is large, and/or (*ii*) there are many paths (the latter is almost always true, especially since our paths may traverse edges in both directions. To mitigate this problem, PathWays includes a *materialized view recommender and view-based rewriting module* (Sec. 2), which significantly improve performance. Thus, PathWays enumerates paths between entities of user-selected types, (*i*) independently of the edge direction, (*ii*) asking for user input to focus on the paths most useful to them, and (*iii*) efficiently. To our knowledge, PathWays is the first system built for this flavor of graph exploration (see also Sec. 4).

2 Demonstration scenarios

PathWays is developed in Java 11, on top of [4,5] which build the data and, respectively, the collection graph (see below), and store them in PostgreSQL.

User interaction with PathWays starts by making some choices: *"Which types of entities to connect?"* (in Fig. 1, organizations and people); *"How many paths to enumerate?"* (say, 20) and *"What is the maximum allowed length for a path?"* (say, 10). In Fig. 1, four paths connect organizations and people; two are shown in yellow and red highlight. Computing the paths on large data graph may be costly. Instead, PathWays leverages a *collection graph* [5], a (much smaller) summary of the data graph, grouping similar data nodes in a single collection node, e.g., the spacecraft nodes 10 and 14 in the data graph are grouped in collection C1. For each collection of nodes having text children, e.g., C8 labeled name, the collection of these text children is denoted, e.g., name#; entities (people, locations, etc.) are extracted from such texts. *Any path in the data graph also exists in the collection graph*, thus PathWays enumerates paths on the smaller collection graph. Each path is then translated in a query over the data graph, to obtain data paths between actual entities. PathWays displays sample entity pairs connected by each path, e.g., Nixon is connected to NASA because he attended a launch involving N. Armstrong. Users can then apply more aggregation/analysis on the entity pairs, look for frequent entities, etc. We will also show how PathWays optimizes path evaluation (see below). We will use real-life datasets, such as PubMed, the NASA dataset, RDF benchmarks, and GeoNames, to investigate connections between people and organizations, e.g., companies funding PubMed article authors, geographic repartition of papers (PubMed) and launches (NASA), etc. A preview of our demonstration can be found at `https://team.inria.fr/cedar/projects/abstra/pathways/`.

3 View materialization and view-based rewriting

When the data graph is large, paths are long, and/or many, evaluating path queries on the graph may be inefficient, despite the graph being extensively in-

dexed. However, as illustrated at right in Fig. 1, paths may *overlap*, e.g., the edges connecting C1, C6 and C7 are common between the two highlighted paths. Leveraging this observation, PathWays identifies the subpaths common to at least two path queries. Then, with the help of a cost model, based on PostgreSQL's estimations, it *materializes the most profitable shared subpath* s_1 where profit is: the decrease in the total path evaluation cost *if the subpath is materialized and its results used to evaluate the paths enclosing it*, minus the cost to materialize the subpath. PathWays then *rewrites* every path query $p_1^1, \ldots, p_1^{n_1}$ containing s_1, using it as a materialized view. Then, we remove $p_1^1, \ldots, p_1^{n_1}$ from the path set, and, in a greedy fashion, again look for the most profitable common subpath s_2 for the remaining paths, etc. We stop when no subpath is profitable (materializing it costs more than its cost savings). The complexity of the above view selection algorithm is $O(N^2L + N^3)$, for N paths of length at most L.

Sample performance saving On the NASA RDF dataset (100.000 triples), we enumerated 100 paths, of length 2 to 8, between locations and people. Evaluating them all took 419 seconds, including 12 that timed-out at 30 seconds. PathWays found 89 common subpaths; 16 were selected by our algorithm, which rewrote 98 path queries using them as materialized views. Materializing the 16 paths took 0.1 second, and the total path query evaluation shrank to 6.93 seconds, a speedup of 60×.

4 Related work and conclusion

Many graph exploration methods exist, see, e.g., [11]. Modern graph query languages such as GPML [7] (no implementation so far) or the JEDI [1] SPARQL extension allow asking for paths between nodes matching some query variables. Other systems interact with users to incrementally build SPARQL queries interesting for them, e.g., [10] for queries with aggregation. In keyword-based search (KBS, in short) [2,4,6,8,12], one asks, e.g., for connections between "Sivel Aliev" (related to the Azeirbadjan president) and "Nice" (where she owns villas). KBS is handy when users *know keywords (entities) to search for*. Complementing the above, PathWays is focused on *identifying and computing all paths between certain extracted entities*, to give a first global look at the dataset content, for graphs obtained from multiple data models. For performance, PathWays leverages a compact graph summary and efficiently materializes views; our view materialization problem, focusing only on paths, is a restriction of that considered, e.g., in [9], enabling a low complexity while being very effective. We are currently adapting our algorithm to other graph data management systems.

Acknowledgments This work has been funded by the DIM RFSI PHD 2020-01 project and the AI Chair SourcesSay (ANR-20-CHIA-0015-01) chair.

References

1. Aebeloe, C., Setty, V., Montoya, G., Hose, K.: Top-K Diversification for Path Queries in Knowledge Graphs. In: ISWC Workshops (2018)
2. Agrawal, S., Chaudhuri, S., Das, G.: DBXplorer: A system for keyword-based search over relational databases. In: ICDE (2002)
3. Anadiotis, A., Balalau, O., Bouganim, T., et al.: Empowering investigative journalism with graph-based heterogeneous data management. IEEE DEBull. (2021)
4. Anadiotis, A., Balalau, O., Conceicao, C., et al.: Graph integration of structured, semistructured and unstructured data for data journalism. Inf. Systems **104** (2022)
5. Barret, N., Manolescu, I., Upadhyay, P.: Abstra: toward generic abstractions for data of any model (demonstration). In: CIKM (2022)
6. Chanial, C., Dziri, R., Galhardas, H., et al.: ConnectionLens: Finding connections across heterogeneous data sources (demonstration). PVLDB **11**(12) (2018)
7. Deutsch, A., Francis, N., Green, A., Hare, K., Li, B., Libkin, L., et al.: Graph pattern matching in GQL and SQL/PGQ. In: SIGMOD (2022)
8. Hristidis, V., Papakonstantinou, Y., Balmin, A.: Keyword proximity search on XML graphs. In: ICDE (2003)
9. Le, W., Kementsietsidis, A., Duan, S., et al.: Scalable multi-query optimization for SPARQL. In: ICDE (2012)
10. Lissandrini, M., Hose, K., Pedersen, T.B.: Example-driven exploratory analytics over knowledge graphs. In: EDBT (2023)
11. Lissandrini, M., Mottin, D., Hose, K., Pedersen, T.B.: Knowledge graph exploration systems: are we lost? In: CIDR. www.cidrdb.org (2022)
12. Yang, J., Yao, W., Zhang, W.: Keyword search on large graphs: A survey. Data Sci. Eng. **6**(2) (2021)