Facilitating Heterogeneous Dataset Understanding

Nelly Barret

Facilitating Heterogeneous Dataset Understanding

Nelly Barret
Inria & Institut Polytechnique de Paris
nelly.barret@inria.fr

ABSTRACT

The era of Big Data and data sharing has lead to very large volumes of data becoming available to users across the world. This data is heterogeneous in its modelling, format and quality. Taking full advantage of such data raises many challenges, in particular related to the integration and the understanding of such data. My PhD thesis, started in January 2021, seeks to develop novel methods to help users without advanced IT skills discover a new dataset, by (*i*) building an abstract understanding of the data, as consisting of *records* and *collections*, (*ii*) interpreting or classifying the data based on users' interests, and leveraging Information Extraction and Natural Language tools.

1 INTRODUCTION

Since the last decades, the Open Data Initiative has led to an increasing number of publicly-spread datasets. Such datasets are often quite large and heterogeneous (depending on the source provider, the field, the kind of data, etc). Many such datasets are large; further, they are extremely heterogeneous, in particular for what concerns their data model (RDF, JSON, XML, CSV, property graphs, relational databases, etc.), their schema (if a schema exists), etc. The scale and heterogeneity make it challenging for human users to identify, among the many available datasets, those that could be used for a given application they have in mind.

This thesis is part of the ConnectionLens project [1], which aims at integrating heterogeneous data into a graph. Our goal is to create small expressive descriptions of what a dataset is about, using the power of integration of ConnectionLens. In this paper, we present the challenges (Section 2), then the approach (Section 3) and finally some preliminary results (Section 4) before concluding (Section 5).

2 CHALLENGES

Finding the right dataset is complicated, especially because they are often not well-documented and it can be difficult to appreciate how it can be useful. Our approach, which aims at helping users to choose a dataset, should satisfy the following requirements:

- **R1: The approach should be applicable to any kind of data.** There are various data formats, such as RDF (as in the Open Data Cloud), but also XML (as in the PubMed database), JSON (most of French open data), relational databases, and so on. This requirement is handled by Section 3.1.
- **R2: The data descriptions we build for users should be sufficiently expressive, but also compact.** Users need to understand what is inside a dataset, but when a full description is complex, we need to bring them only the most important facts about it. We discuss how to fulfil this requirement in Sections 3.2 and 3.3.

3 APPROACH

ConnectionLens is a system capable to produce a graph G from any dataset of any format, where each node is a piece of data and edges link these nodes to reflect the content of the original source. Moreover, an entity extraction process is applied on text nodes, to extract from them named entities, such as `Person`, `Location`, `Organization`, `Date`, etc. In my thesis, to be able to produce compact descriptions of any data format, we leverage ConnectionLens to start our summarization method from the graph G. Our approach is the following:

(1) **Build a structural summary of G.** The structural summary G' is a graph computed out of G, potentially much smaller than G, and which gives us a first idea of groups of nodes that may contain similar information: each such group of G nodes is *represented* by a single node in G'.
(2) **Find collections and records.** Starting from the summary, we seek to identify the nodes that represent *records*, that is, objects of a certain "kind" with some internal structure, and *collections*, that is, containers of potentially many records of the same "kind".
(3) **Categorize collections.** Finally, we aim at *classifying collections* among a set of categories $\mathcal{K}$, containing (*i*) the kind(s) of data that the user is looking for, if the user can formulate such a request, e.g., "Books", or "Places to visit", and/or (*ii*) a set of generic categories we pre-define, such as `Person`, `Organization`, `Location`, `Event` and `Creative work`. The categorization adds a limited form of semantics (we keep things simple on purpose since we assume non-technical users), and enable adapting to the users' interest.

3.1 Summarization

We explain now how we compute the summary G' of G. For efficiency, we distinguish two cases: rooted, acyclic data source graphs, vs. the general case where graphs may have cycles and/or may not have a root.

Rooted acyclic graphs. These graphs are obtained for instance from XML or JSON datasets. On such graphs, we apply the strong DataGuide summarization method [4] to create G' from G. A Dataguide is a concise summary of the structure of a database. This method builds a set of paths, such that each path of the DAG appears exactly once in the summary. Such summarization method works only on acyclic graphs because the recursion should not encounter a cycle.

General graphs. Such graphs can originate in RDF, property graphs, or relational database datasets (where primary-foreign keys can lead to cyclic connections between the tuples). For such graphs, we need a graph summarization method that (*i*) reflects all the graph, (*ii*) groups nodes into equivalence classes and (*iii*) can be computed efficiently even from large graphs. RDFQuotient [3], originally introduced for RDF but easy to adapt to arbitrary graphs, meets these criteria, thus we rely on it to compute the summary G' of G for non-acyclic graphs. RDFQuotient gives a set of equivalence classes between nodes based on their types and their properties.

3.2 Records and Collections

We seek to understand G' based on two key concepts:

- A **Record** is basically a *thing*; in data modelling terms, it describes either an entity or a relationship. It has some properties (e.g. a `title` and a `DOI` for a paper) and can handle nested collections (e.g. the authors list of a paper).
- A **Collection** is a set of similar records (e.g. a bibliography is a collection of books). They are *explicit* when a node handles similar records; or *implicit* when some records refer to the same purpose without being handled by a node.

Other nodes in G' are called Sub-Records and are mainly the properties of the records (i.e. the set of outgoing properties of a record r, referred as $r.\mathcal{P}$). Furthermore, we compute the *signature* of each sub-record s, where the signature is compound of a *domain* ("*to which categories s belongs to?*") and a *range* ("*to which categories s points to?*"). For example, the sub-record `settledDownIn` has for domain {`Person, Organization`} and for range {`Location`}.

To find them, we first determine collections and then, in a top-down fashion, the direct children of collections are identified as records. To compute collections, we rely on a clustering algorithm we devised, based on the *support* of a set of properties among a set of potential records (how many of these records have this set of properties). Our clustering algorithm identifies both *explicit* collections, where a G' node is actually the parent of all the nodes representing the records in the collection, and *implicit* collections, where such a common parent/collection node does not exist in G'.

3.3 Analysis and Categorization of Collections

Given a set of hints $\mathcal{H}$ and a set of user-defined categories $\mathcal{K}$, we aim at categorizing a collection c among $\mathcal{K}$, i.e. give a category $k \in \mathcal{K}$ to c using $\mathcal{H}$, as illustrated by Algorithm 1. A **hint** h is a triple $\langle A, l, B\rangle$ where A is the *domain* $\subseteq \mathcal{K}$, l is the label and B is the *range* $\subseteq \mathcal{K}$. For instance, the hint ⟨Organization, hasCEO, Person⟩ states that a collection having a record holding the property `hasCEO`, whose signature's range matches `Person`, should be categorized as an `Organization`.

For each record $r \in c$, we initialize $\mathcal{K}_r$ (set of candidate categories in which r may belong) and scores (score of nc for each hint in $\mathcal{H}$). Then, if r has a label semantically close to one of the category in $\mathcal{K}$, this category is stored as a candidate category in $\mathcal{K}_r$. For each child $nc \in r$, we create a pair π containing the label and the signature of nc. Then, we compute the similarity of π with each hint h in $\mathcal{H}$, where the similarity is based on the label and the signature of both elements. We choose the hint h leading to the highest similarity score for each π. Each category indicated by the domain of h gets a vote. Then, we classify r in the category that gets the highest number of votes or `Other` if no category is frequent enough. Finally, we classify c in the most represented category in its records. We also determine if a collection describes entities or relationships, by looking at the connections between the collections.

4 STATUS

We have fully implemented our approach in a prototype, which leverages the graph creation and storage of ConnectionLens [1], and includes the novel algorithms described in Section 3. More details can be found in a short paper [2].

Figure 1 shows an example of our approach applied on a set of PubMed articles. The set of articles is considered as a collection of `Creative Work`. Moreover, the authors are identified as a collection of `Persons`.

Algorithm 1: Classifying a collection c

Input: a collection c, hints $\mathcal{H}$, categories $\mathcal{K}$
Output: a category $k \in \mathcal{K}$ or `Other`

foreach $r \in C$ **do**
  $\mathcal{K}_r \leftarrow \emptyset$
  scores $\leftarrow \emptyset$
  foreach $k \in \mathcal{K}$ **do**
    if *the similarity between k and the label of r is higher than a threshold* **then**
      $\mathcal{K}_r \leftarrow \mathcal{K}_r \cup \{k\}$
  foreach $nc \in r.children$ **do**
    $\pi \leftarrow (nc.label, nc.signature)$
    foreach $h \in \mathcal{H}$ **do**
      scores $\leftarrow$ scores $\bigcup(h, sim(\pi, h))$
    bestHint $\leftarrow$ argmax(scores)
    $\mathcal{K}_r \leftarrow \mathcal{K}_r \cup \{$ bestHint.domain $\}$
  Classify r in the most frequent $k \in \mathcal{K}_r$, or `Other`
Classify c with the most frequent category of its records

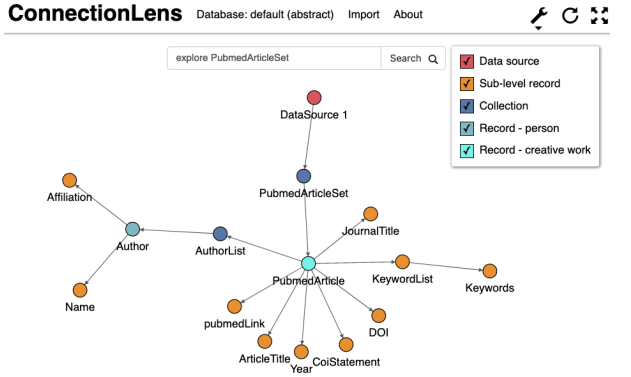

Figure 1: Example of G', an abstract graph with collections and categorized records.

5 CONCLUSION AND PERSPECTIVES

My PhD thesis aims to create expressive descriptions of big heterogeneous datasets by using summarization methods and categorization of expressive structures (records and collections). Beyond finalizing the implementation of our platform for all the data models we consider (notably, beyond XML and RDF, also JSON and property graphs), we will experiment to analyse its scalability as well as the expressiveness and precision of the record categorization. Next, we will investigate the adoption of sampling-based approaches, to try to construct such dataset descriptions without traversing the dataset entirely, in order to further improve performance.

Thesis context My PhD is funded by DIM RFSI and is a collaboration between Inria and WeDoData, a SME specialized in data visualization and interactive data-driven Web content. My PhD advisers are Ioana Manolescu (Inria) and Karen Bastien (WeDoData).

Acknowledgments. This work is funded by DIM RFSI PHD 2020-01 and AI Chair SourcesSay project (ANR-20-CHIA-0015-01) grants.

REFERENCES

[1] A. C. Anadiotis, O. Balalau, C. Conceicao, H. Galhardas, M. Y. Haddad, I. Manolescu, T. Merabti, and J. You. Graph integration of structured, semistructured and unstructured data for data journalism. *Information Systems*, July 2021.
[2] N. Barret, I. Manolescu, and P. Upadhyay. Toward generic abstractions for data of any model. 2021. Short paper, accepted for publication at BDA 2021.
[3] F. Goasdoué, P. Guzewicz, and I. Manolescu. RDF graph summarization for first-sight structure discovery. *The VLDB Journal*, 29(5), Apr. 2020.
[4] R. Goldman and J. Widom. Dataguides: Enabling query formulation and optimization in semistructured databases. In *VLDB*, 1997.