A Data-Driven Graph Schema

Larry González[1] and Aidan Hogan[2]

[1] Center for Advancing Electronics Dresden (cfaed) TU Dresden, Germany.
larry.gonzalez@tu-dresden.de
[2] Millennium Institute for Foundational Research on Data, DCC, University of Chile.
ahogan@dcc.uchile.cl

Abstract. In this paper, we summarise our results on Modelling Dynamics in Semantic Web Knowledge Graphs published at WWW 2018 where we proposed a novel data-driven schema for graphs and apply it for the use-case of predicting high-level changes in Wikidata.

1 Introduction

Graph-based data models [1] have become increasingly common in data management scenarios that require flexibility beyond what is offered by traditional relational databases. Such flexibility is particularly important in Web scenarios, where potentially many users may be involved (either directly or indirectly) in the creation, management, and curation of data. An example of such a scenario is the Wikidata knowledge graph [2] where users can add new properties and types that can be used to define further data.

The flip-side of flexibility is higher levels of heterogeneity. Conceptually understanding the current state of a knowledge graph – in terms of what data it contains, what it is missing, how it can be effectively queried, what has changed recently, etc. – is thus a major challenge: it is unclear how to distil an adequate, high-level description that captures an actionable overview of knowledge graphs.

We thus need well-founded methodologies to make sense of knowledge graphs, where an obvious approach is to define some notion of *schema* for such graphs. The traditional approach in the Semantic Web has been what Pham and Boncz [3] call the *schema first* approach, which defines the schema that the data should follow. The most established language for specifying such schemas is RDFS. An alternative to the *schema first* approach is the *schema last* approach [3], which foregoes an upfront schema and rather lets the data evolve naturally; thereafter, the goal is to understand what the legacy graph data contain by extracting high-level summaries that characterise the graph, resulting in a *data-driven schema.*

In this paper, we summarise recently published results [4] on a novel approach to compute a data-driven schema from knowledge graphs. We believe that such schemas are useful for understanding what a knowledge graph contains, and how it can be queried, among several other use-cases. Nevertheless, in this work we focus on the use-case of predicting how the knowledge graph will evolve in future versions, which could be used for measuring the time-to-live of cached SPARQL results, identifying missing properties for entities, etc.

2 Preliminaries

RDF is a graph-structured model based on three disjoint sets of terms: IRIs ($\mathbf{I}$), literals ($\mathbf{L}$) and blank nodes ($\mathbf{B}$). Claims involving these terms can be organised into *RDF triples* $(s, p, o) \in (\mathbf{I} \cup \mathbf{B}) \times \mathbf{I} \times (\mathbf{I} \cup \mathbf{B} \cup \mathbf{L})$, where s is called *subject*, p is called *predicate*, and o is called *object*. An *RDF graph* G is then a finite set of RDF triples; each such triple $(s, p, o) \in G$ can be viewed as a directed labelled edge of the form $s \xrightarrow{p} o$. The terms used in the predicate position are referred to as *properties*. We use the term *entity* to refer to the real-world objects identified by the subjects of the graph. Given an RDF graph G, for $\bullet \in \{\mathrm{S}, \mathrm{P}, \mathrm{O}\}$, we denote by $\pi_\bullet(G)$ the projection of the set of terms appearing in a particular triple position in G; e.g., $\pi_{\mathrm{S}}(G) := \{s \mid \exists p, o : (s, p, o) \in G\}$. We also use this notation for more than one triple position, for example, $\pi_{\mathrm{S,P}}(G) := \{(s, p) \mid \exists o : (s, p, o) \in G\}$.

3 A Data-Driven Schema for (RDF) Graphs

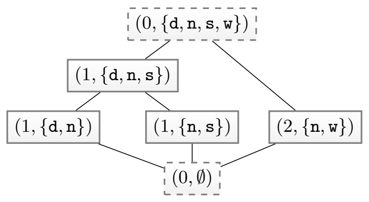
To define our data-driven schema proposal, let $[\![G]\!] \subseteq 2^{\pi_{\mathrm{S}}(G)} \times 2^{\pi_{\mathrm{P}}(G)}$ denote a set such that $\bigcup_{(S,P)\in[\![G]\!]} S \times P = \pi_{\mathrm{S,P}}(G)$, and where for all $(S, P) \in [\![G]\!]$, it holds that $S \neq \emptyset$, $P \neq \emptyset$, and there does not exist $(S', P') \in [\![G]\!]$, $(S, P) \neq (S', P')$ such that $S \cap S' \neq \emptyset$ or $P = P'$. Intuitively, letting $[\![s]\!]_G := \{p \mid \exists o : (s, p, o) \in G\}$ denote the *characteristic set* of s in G [5], then each pair $(S, P) \in [\![G]\!]$ represents the (non-empty) set of all subjects S with the (non-empty) *characteristic set* P. We further define $[\![P]\!]_G = S$ such that $(S, P) \in [\![G]\!]$ (or $[\![P]\!]_G = \emptyset$ if no such S exists), and $[\![S]\!]_G = P$ such that $(S, P) \in [\![G]\!]$ (or $[\![S]\!]_G = \emptyset$ if no such P exists). Abusing notation, we say that $(S, P) \subseteq (S', P')$ iff $P \subseteq P'$. We then also define $[\![G]\!]^* := [\![G]\!] \cup \{(\emptyset, \emptyset), ([\![\pi_{\mathrm{P}}(G)]\!]_G, \pi_{\mathrm{P}}(G))\}$, adding a bottom and top concept (if needed) respectively in the $\subseteq$ order. Finally, for a graph G, our data-driven schema proposal is then given by the lattice $\mathcal{L} = ([\![G]\!]^*, \subseteq)$.

Given that large-scale knowledge graphs may often have orders of magnitude more subjects than predicates, we can greatly reduce the overall (e.g., in-memory) size of the lattice by encoding the number of subjects rather than the full set of subjects. In other words, given a lattice $\mathcal{L} = ([\![G]\!]^*, \subseteq)$, we define $[\![G]\!]^\# := \{(n, P) \mid \exists S : (S, P) \in [\![G]\!]^*, n = |S|\}$ with lattice $\mathcal{L}^\# := ([\![G]\!]^\#, \subseteq)$. We denote by $[\![P]\!]^\#_G = |[\![P]\!]_G|$ the number of subjects that P has. Figure 1 provides an example RDF graph and the Hasse diagram for its corresponding $\mathcal{L}^\#$.

4 Lattice Diff-Algebra

Though we believe that the lattices defined previously may satisfy a number of applications, we currently focus on the use-case of modelling and predicting changes in a graph. More specifically, if we have the lattices for two versions of a knowledge graph, we can apply a diff to see high-level changes between both versions. Furthermore, given such a diff, we could further consider adding that diff to the most recent version to try predict future changes.

```
:UT :name "U Thurman" ; :star :Gattaca .
:GO :name "G Orwell" ; :writer :1984 .
:AK :name "A Kurosawa" ; :director :Ikiru , :Ran .
:PD :name "PK Dick" ; :writer :Ubik , :Valis .
:CE :name "C Eastwood" ; :director :Sully ;
    :star :Unforgiven , :Tightrope .
```



$(0, \{\mathtt{d}, \mathtt{n}, \mathtt{s}, \mathtt{w}\})$
$(1, \{\mathtt{d}, \mathtt{n}, \mathtt{s}\})$
$(1, \{\mathtt{d}, \mathtt{n}\})$
$(1, \{\mathtt{n}, \mathtt{s}\})$
$(2, \{\mathtt{n}, \mathtt{w}\})$
$(0, \emptyset)$

Fig. 1: Example RDF graph and corresponding data-driven schema where characteristic sets are annotated with the number of subjects (#-lattice). Properties are abbreviated by their initial: `d`/`:director`, `n`/`:name`, `s`/`:star`, `w`/`:writer`

Defining lattice diffs Let $\mathcal{L}_i = (\llbracket G_i \rrbracket^*, \subseteq)$ and $\mathcal{L}_j = (\llbracket G_j \rrbracket^*, \subseteq)$ be the lattices for two versions (i and j) of an RDF graph G. We define the diff between these two lattices as $\Delta_{j,i} := \{(\llbracket s \rrbracket_{G_j}, s, \llbracket s \rrbracket_{G_i}) \mid s \in \pi_{\mathrm{S}}(G_i \cup G_j)\}$; note that $\llbracket s \rrbracket_{G_j} = \emptyset$ for deleted subjects and $\llbracket s \rrbracket_{G_i} = \emptyset$ for new subjects. Given $\Delta_{j,i}$, we also define a cardinality-only version $\Delta^{\#}_{j,i} := \{(P', n, P) : n = |\{s : (P', s, P) \in \Delta_{j,i}\}|\}$, where by $\Delta^{\#}_{j,i}(P', P)$ we denote n such that $(P', n, P) \in \Delta^{\#}_{j,i}$ or 0 if no such n exists.

Predicting future #-lattices Given $\Delta^{\#}_{j,i}$, and $\mathcal{L}^{\#}_k$ (for k a third version of the graph), we can "add" the changes between the i^{th} and j^{th} versions to the k^{th} version to predict the $(k+j-i)^{\text{th}}$ version (where typically $i < j \leq k$). We will thus define the operation $\mathcal{L}^{\#}_k + \Delta^{\#}_{j,i}$ as producing a #-lattice $\mathcal{L}^{\#}_{k,j,i}$ predicting $\mathcal{L}^{\#}_{[k+j-i]}$. To apply this operation we consider the ratio of subjects moving from a source to a target characteristic set. Formally, we define the ratio of subjects of P (where $\llbracket P \rrbracket^{\#}_{G_i} \neq 0$, $P \neq \emptyset$) moving to P' (where $P' \neq \emptyset$) as $\rho_{j,i}(P', P) := \frac{\Delta^{\#}_{j,i}(P', P)}{\llbracket P \rrbracket^{\#}_{G_i}}$; in the case that $\llbracket P \rrbracket^{\#}_{G_i} = 0$, we define $\rho_{j,i}(P', P) = 1$ iff $P' = P$, or 0 otherwise. We then define $\mathcal{L}^{-}_{k,j,i} := (\{(\sigma(P), P) \mid P \neq \emptyset \text{ and } \sigma(P) \neq 0\}, \subseteq)$, where:

$$\sigma(P) := \mathrm{round}\left(\sum_{P_k \in \{P \mid \exists S : (S, P) \in \llbracket G_k \rrbracket\}} \rho_{j,i}(P, P_k) \times \llbracket P_k \rrbracket^{\#}_{G_k} \right) + \Delta^{\#}_{j,i}(P, \emptyset) \,.$$

The summand $\Delta^{\#}_{j,i}(P, \emptyset)$ adds the number of fresh subjects (not appearing in version i) added to P in version j. Finally, we add top and bottom concepts (as before) to $\mathcal{L}^{-}_{k,j,i}$ to generate the predicted #-lattice $\mathcal{L}^{\#}_{k,j,i}$.

5 Evaluation

We consider 11 weeks of "truthy" RDF dumps of Wikidata from 2017-04-18 to 2017-06-27; the first version has 1,102,242,331 triples, 54,236,592 unique subjects and 3,276 unique properties, while the last version has 1,293,099,057 triples

(+17%), 57,197,406 unique subjects (+5%) and 3,492 unique properties (+6%). From the last version, with a MapReduce implementation, we extract 2,118,109 characteristic sets in approximately 2.5 hours; computing the lattice by the $\subseteq$ relation then took almost 8 hours on a single machine [4].

To test the quality of the future #-lattices we predict – specifically the number of subjects per characteristic set in the future unseen version – we run experiments where we consider from 2–5 previous weekly versions to predict the next version of the #-lattice. As a baseline, for each characteristic set, we apply linear regression over the number of subjects in that characteristic set for the previous weeks to predict the number of subjects for the next week; we compare this baseline with our diff algebra ($\varDelta$), computing the error with respect to the real lattice of the predicted week. The results are available in [4], where we show that our diff algebra outperforms the linear regression baseline method in all cases; we believe that this is because our diff algebra considers the number of subjects remaining in source characteristic sets for its predictions whereas the baseline does not consider where predicted new subjects will come from.

6 Conclusion

We have proposed a form of data-driven schema for large-scale knowledge graphs and shown it to be feasible to compute. As a concrete use-case, we presented an algebraic method by which these schemas can be used to predict high-level changes in the dataset. Our evaluation over 11 weeks of Wikidata demonstrates that such predictions are feasible to compute; furthermore, we validated the quality of predictions made by our algebraic approach against a linear-model baseline. We refer the interested reader to the full paper [4] for additional details on the proposed schema, examples of diffs, algorithms to compute the lattices, statistics on the lattices produced, details of the experiments, and further discussion.

Acknowledgements: This work was supported by the Millenium Scientific Initiative, by Fondecyt Grant No. 1181896 and by the German Research Foundation (DFG) in CRC 912 (HAEC) and in Emmy Noether grant KR 4381/1-1.

References

1. Angles, R., Arenas, M., Barceló, P., Hogan, A., Reutter, J., Vrgoč, D.: Foundations of modern query languages for graph databases. ACM Comp. Surveys **50**(5) (2017)
2. Vrandecic, D., Krötzsch, M.: Wikidata: a free collaborative knowledgebase. Commun. ACM **57**(10) (2014) 78–85
3. Pham, M., Boncz, P.A.: Exploiting Emergent Schemas to Make RDF Systems More Efficient. In: ISWC. Lecture Notes in Computer Science, Springer (2016) 463–479
4. González, L., Hogan, A.: Modelling dynamics in semantic web knowledge graphs with formal concept analysis. In: WWW. (2018)
5. Neumann, T., Moerkotte, G.: Characteristic sets: Accurate cardinality estimation for RDF queries with multiple joins. In: ICDE, IEEE Comp. Society (2011) 984–994