Augmenting Logic-based Knowledge Graphs: The Case of Company Graphs

Paolo Atzeni[1], Luigi Bellomarini[2], Michela Iezzi[2],
Emanuel Sallinger[3,4], and Adriano Vlad[4]

[1] Università Roma Tre
[2] Banca d'Italia
[3] TU Wien
[4] University of Oxford

1 The Industrial Setting

Company ownership graphs are central objects in corporate economics [4, 10, 13, 17] and are of high importance for central banks to solve problems in *banking supervision*, *credit-worthiness*, *anti-money laundering* and many more areas. The Bank of Italy owns the database of Italian companies that contains details about shareholding structures. In a graph-based view of the database, as shown in Figure 1, we see *ownership* as the core concept: nodes are companies and persons (black resp. blue nodes), and ownership edges (black solid links) are labelled with the shares a company or person x owns of a company y.

An important problem with company graphs is *company control* [9], which amounts to deciding whether a company x controls a company y, i.e., x can force decisions of y. A company (or a person) x *controls* a company y, if: (i) x directly owns more than 50% of y; or, (ii) x controls a set of companies that jointly, and possibly together with x, own more than 50% of y.

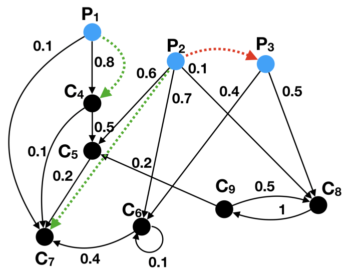

Fig. 1. Sample from the Italian graph.

Besides financial relationships, *personal or family connections* (of various degrees, e.g., “PartnerOf”, “SiblingOf”, and so on) enable a much broader use of such company graphs in several fields such as anti-money laundering, fraud detection or economic and statistical research [12]. In these fields, the definition of control can be extended to families. For instance, in Figure 1: P_1 *controls* C_4 *with a direct* 80% *edge;* P_2 *controls* C_7*, via* C_5 *and* C_6*. Dashed green edges represent control links. The dashed red edge between* P_2 *and* P_3 *represents a “PartnerOf” relationship.* P_2 *and* P_3 *are in the same family. Neither* P_2 *nor* P_3 *control* C_8*, but their family jointly owns* 60% *and thus controls* C_8. In this paper we continue our recent work [2] and study the problem of predicting family links as a special case of a wider class of *graph augmentation problems* providing a hybrid embedding-reasoning approach to it.

2 The Vada-Link Framework

With *knowledge graph augmentation*, (KG augmentation) we wish to characterize a special case of *link prediction* [15], concerned with predicting hidden links in network structures. Unlike the standard problem, our emphasis is on the need for a careful combination of the extensional data (existing nodes and edges, namely the *ground extensional component*) and the available domain knowledge (the *intensional component*). This view on the problem is well captured by *logic-based* Knowledge Graphs [6], which represent available data as facts, often encoding *property graphs* (PGs) [1] in a relational form, and domain knowledge as reasoning rules. Rules are modeled in some logic formalism, e.g., Vadalog [7] a language in the Datalog$^\pm$ [8] family, striking a good balance between computational complexity and expressive power. The application of rules upon the extensional component allows to accomplish complex *reasoning tasks* by generating new entailed facts and thus new knowledge.

At the core of our approach, Vada-Link, there is the idea of modeling KG augmentation problems as *reasoning tasks* on logic-based KGs, having enterprise data stores as extensional component (typically a PG) and Vadalog reasoning rules expressing the augmentation logic. Link prediction, and therefore KG augmentation, is an intrinsically quadratic problem, making naïve approaches inapplicable to dense graphs like that of the Italian companies with millions of nodes. Inspired by entity resolution approaches [11], Vada-Link addresses scalability with a principled interplay of *node embeddings*-based *clustering* [14] that reduces the set of candidate nodes, and pure logic reasoning, which exhaustively determines links inside each cluster and hence improves clustering accuracy.

The Vadalog reasoning rules for KG augmentation are structured into three sets: 1. *input mappings*: taking as input the relational representation of one specific PG, and transforming it into higher-level concepts: generic nodes, generic edges and types and properties for those nodes and edges; 2. *link prediction logic*: containing the core reasoning process giving rise to new edges; 3. *output mappings*: transforming the high-level generic links that have been created by the link prediction logic into their relational representation in the PG.

Let us come to the link prediction logic, described by the following rules:

$$\begin{gathered}
(1)\ \mathrm{Node}(x, f_1^x, \ldots, f_n^x), \mathrm{Link}(e, v, w, f_1^e, \ldots, f_m^e),\\
\mathrm{NodeType}(x, t_n), \mathrm{EdgeType}(e, t_e),\\
b_1 = \#\mathrm{GraphEmbedClust}(f_1^x, \ldots, f_n^x, f_1^e, \ldots, f_m^e, t_n, t_e, \langle e \rangle),\\
b_2 = \#\mathrm{GenerateBlocks}(f_1^x, \ldots, f_n^x, t_n) \rightarrow \mathrm{Block}(b_1, b_2, x)\\
(2)\ \mathrm{Node}(x, f_1^x, \ldots, f_n^x), \mathrm{Node}(y, f_1^y, \ldots, f_n^y),\\
\mathrm{NodeType}(x, t_n), \mathrm{NodeType}(y, t_n), x \neq y,\\
\mathrm{Block}(b_1, b_2, x), \mathrm{Block}(b_1, b_2, y), \mathrm{LinkClass}(\mathrm{t}),\\
\mathrm{Candidate}(x, y, t) \rightarrow \exists z\ \mathrm{Link}(z, x, y, \ldots), \mathrm{EdgeType}(z, t).
\end{gathered}$$

For every node x, Rule (1) considers all edges e and positions x as a two-level nested clustering structure represented by the atom `Block`, where b_1 and b_2 are the clustering levels. The first clustering is established by applying the function #GraphEmbedClust, which wraps a function call to a specific clustering algorithm based on a node embedding primitive, e.g., node2vec [14]. It takes as input the features of x, the edges e of the graph along with their features, and the respective types t_n and t_e, and returns the identifier b_1 of the first-level cluster. #GraphEmbedClust is a *monotonic aggregation function.*

The second-level clustering is determined by applying the function #GenerateBlocks, whose resulting cluster identifier b_2 only depends on the node properties and type. #GenerateBlocks is a *fact-level function*: for a specific binding of the function arguments, a value for b_1 is produced. The function is in some sense polymorphic: depending on the type t_n of the involved nodes, a specific semantics is applied to decide the target cluster on the basis of the node features.

For every second-level cluster defined by a `Block` fact, Rule (2) exhaustively considers all pairs of nodes x and y and every possible `LinkClass` t (of which with respect to our case many exist: Control, ParentOf, PartnerOf, etc.). The `Candidate` predicate is used to decide whether a `Link` from x to y must be produced or not. If this is the case, a new t-typed edge is created. Rule (2) compares only the pairs of nodes in the same sub-cluster (identified by b_1 and b_2). `Candidate` is defined for company control (3-4) and family links (5) as:

$$\begin{gathered}
(3)\ \mathrm{Node}(x, f_1, \ldots, f_n), \mathrm{NodeType}(x, \mathrm{Company}) \rightarrow \mathrm{Candidate}(x, x, \mathrm{Control}).\\
(4)\ \mathrm{Candidate}(x, z, \mathrm{Control}), \mathrm{Link}(u, z, y, w)\mathrm{EdgeType}(\mathrm{u}, \mathrm{Shareholding}),\\
\mathit{msum}(w, \langle z \rangle) > 0.5 \rightarrow \mathrm{Candidate}(x, y, \mathrm{Control}).\\
(5)\ \mathrm{Node}(x, f_1^x \ldots f_n^x), \mathrm{Node}(y, f_1^y \ldots f_n^y), \mathrm{NodeType}(\mathrm{x}, \mathrm{Person}),\\
\mathrm{NodeType}(\mathrm{y}, \mathrm{Person}), \#\mathrm{LinkProbability}(\mathrm{f_1^x \ldots f_n^x}, \mathrm{f_1^y \ldots f_n^y}) > \mathrm{T} \rightarrow\\
\mathrm{Candidate}(\mathrm{x}, \mathrm{y}, \mathrm{PartnerOf}).
\end{gathered}$$

Many different models to predict family links exist. As common in these cases, we simply define the presence of a family link between x and y with a multi-feature Bayesian classifier (wrapped by #LinkProbability).

Discussion. Our approach is *schema independent* as Vada-Link is able to perform KG augmentation regardless of the specific input PGs. We implement it with polymorphic `Candidate` atoms for each problem. The approach is *model independent* because the extensional component can originate from heterogeneous data sources, even based on different data models. Finally, we adopt a kind of *reinforcement principle* because the the first-level clustering in the Vada-Link algorithm is gradually improved with new edges from Rule (2).

Termination is guaranteed as the number of `Link`s that can be generated by Rule (2) is finite and, in the worst case, it amounts to $|N|^2 \times C$, where N are the PG nodes and C is the number of possible link types. Rule (1) produces a finite number of clusterings $\langle b_1, b_2 \rangle$, since it can fire in the worst case for every single edge in E plus all the ones introduced by Rule (2).

Our solution is *scalable*, given Vadalog tractability [7], and limits the search space via multi-level clustering. In the average case, clustering allows for linear behaviour. In the worst case, the approach performs $|N|^2 \times C$ comparisons, (N are the nodes and C is the number of possible link types). However, for high density, complexity is dominated by embedding algorithms, having quadratic complexity in the branching factor [16].

3 Experiments

We provide an experimental evaluation of Vada-Link to family detection, highlighting its scalability and accuracy for both real-world and artificial settings.

Datasets. For the *real-world case*, we use the database of Italian companies of Banca d'Italia whose details can be found in [2]. For the *synthetic case*, we adopt Barábasi algorithm [3] to generate realistic scale-free company networks with 6-feature nodes obtained respecting the original statistical properties.

Software and hardware configuration. We ran the experiments on a MacBook with 1.8 GHz Intel Core i5 and 4 GB 1600 MHz DDR3 memory. The clustering functions have been executed with Python 3.2.3.

3.1 Evaluating Scalability

We tested the scalability of Vada-Link for different clustering structures, varying graph topology (e.g., number of nodes, density) and data distribution.

Varying number of nodes. We built 20 scenarios with subsets ($\approx$ 1-100k nodes) from the Italian company graph and averaged elapsed times. To further stress the system, we also built 6 artificial graphs of higher density but same size and scale-free topology as the real-world ones. Figure 2(a) and 2(c) show Vada-Link good scalability for the real-world and synthetic case respectively. The trend is linear and the approach very effective.

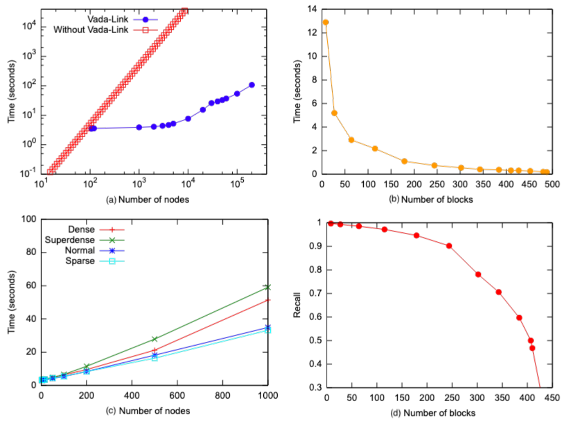

Fig. 2. Vada-Link execution time and recall.

Varying the number of clusters. We crafted a *real-world-like* experiment, using the Italian company graph. Vada-Link clustering technique recursively and jointly employs node2vec (first-level clustering) and feature based-blocking (second-level clustering); using a deterministic mapping, we assign –via hashing or Skolem functors– a feature vector $f_1, \ldots, f_n$ into a second-level cluster identifier. In this experiment, by artificially tweaking the value of k of such n features, we hijack the mapping into an increasing number of clusters of decreasing size and observe how this affects elapsed time. We extract values for the vector $f_1, \ldots, f_k$ from a discrete multivariate uniform distribution over the sample space $S_1, \ldots, S_k$. By restricting (expanding) the cardinality of the domain of $S_1 \times \ldots \times S_k$, we induce more (fewer) and smaller (bigger) clusters. Specifically, we mapped $f_1, \ldots, f_n$ into $\approx$ 1-500 clusters.

Figure 2(b) confirms that the effective application of VADA-LINK requires a careful feature engineering phase: the number and size of the clusters vary according to the feature selectivity. For example, searching for the "siblingOf" relationship among people with common last names and same age range would lead to large clusters. VADA-LINK, through the joint adoption of a hybrid node-2vec/feature-based clustering and recursive self-improving approach, makes it easier to balance the number of clusters, elapsed time, and recall.

Varying the density. We built 4 artificial graph scenarios, *superdense*, *dense*, *normal*, *sparse* and measured the relevant execution time. Figure 2(c) shows the impact of graph density on the performance of VADA-LINK. Elapsed times increase significantly for more than 500 nodes whilst, for lower values, sparse, normal and dense show similar trends. On the contrary, the computation for superdense graphs is slower taking ≈30 seconds for 500 nodes. The trend is amplified for greater values, with superlinear growth for dense and superdense. While for second-level clustering #GenerateBlocks is not affected by node density, since it only considers node features, both first-level clustering (#GraphEmbedClust) and the implementations of `Candidate` predicate are. Node2vec processes a number of random walks that grows with the density; nevertheless, once a density is fixed, it scales almost linearly with the number of nodes, as also experimentally shown in [14]. Also the behaviour of `Candidate` is highly dependent on the considered KG augmentation problem. For example, the detection of family connections has good scalability with respect to density, as evident in Figure 2(c). Company control and close link detection are more challenging (specific experiments in [7, 5]). Nevertheless, thanks to clustering, VADA-LINK can achieve good behaviour also in this case, clearly at the cost of loss in accuracy.

3.2 Evaluating Accuracy

Following a common validation approach for link prediction settings [14], we arbitrarily remove some edges from the initial graph, then try to predict and recover them, and evaluate the overall tradeoff between scalability and recall.

Varying the number of clusters. We built 10 realistic random graphs S_i (with $1 \leq i \leq 10$). For each of them, we ran VADA-LINK concentrating all nodes in one single cluster and producing all the theoretically possible links resulting in an augmented one $\hat{S}_i$. Then, from each $\hat{S}_i$, we randomly selected 10 edge sets Θ_{ij} (with $1 \leq j \leq 10$), each containing 20% of the predicted links and generated new subgraphs $S^{\Theta_{ij}}$ without those edges. For each $S^{\Theta_{ij}}$ we ran VADA-LINK by varying the number of clusters with 20 configurations from 1 to 500. For each cluster, we obtained an augmented graph $\hat{S}^{\Theta_{ij}}$ and the recall R_{ijc} as $|E(\hat{S}^{\Theta_{ij}})|/|E(\hat{S}_i)|$, i.e., the percentage of removed edges that have been recovered. By comparing Figures 2(b) and 2(d), we observe that 10 clusters enable processing times under 10 seconds and an effective efficiency-recall balance. The recursive interplay between first- and second-level clustering compensates for increases in the number of clusters and contributes to a favourable balance between scalability and recall proving the robustness of VADA-LINK.

4 Conclusion

In this paper, we proposed Vada-Link, a novel approach for weaving enterprise KGs by discovering hidden links. At the core of our technique, there is a principled formulation of the problem as a Vadalog reasoning task, which guarantees scalability and problem independence, as well as machine learning/embeddings.

Acknowledgements. We acknowledge the support of the WWTF grant VRG18-013, the EPSRC grant EP/M025268/1, and the EU Horizon 2020 grant 809965.

References

1. Renzo Angles. The property graph database model. In *AMW*, 2018.
2. Paolo Atzeni, Luigi Bellomarini, Michela Iezzi, Emanuel Sallinger, and Adriano Vlad. Weaving enterprise knowledge graphs: The case of company ownership graphs. In *EDBT*, pages 555–566. OpenProceedings.org, 2020.
3. Albert-Laszlo Barabasi and Reka Albert. Emergence of scaling in random networks. *Science (New York, N.Y.)*, 286:509–12, 11 1999.
4. Fabrizio Barca and Marco Becht. The control of corporate europe. oxford university press, european corporate governance network. 2001.
5. Luigi Bellomarini, Davide Benedetto, Georg Gottlob, and Emanuel Sallinger. Vadalog: A modern architecture for automated reasoning with large knowledge graphs. *Information Systems*, page 101528, 2020.
6. Luigi Bellomarini, Daniele Fakhoury, Georg Gottlob, and Emanuel Sallinger. Knowledge graphs and enterprise AI: the promise of an enabling technology. In *ICDE*, pages 26–37. IEEE, 2019.
7. Luigi Bellomarini, Emanuel Sallinger, and Georg Gottlob. The vadalog system: Datalog-based reasoning for knowledge graphs. *PVLDB*, 11(9):975–987, 2018.
8. Andrea Calì, Georg Gottlob, Thomas Lukasiewicz, Bruno Marnette, and Andreas Pieris. Datalog+/-: A family of logical knowledge representation and query languages for new applications. In *LICS*, 2010.
9. Stefano Ceri, Georg Gottlob, and Letizia Tanca. *Logic programming and databases*. Springer, 2012.
10. Ariane Chapelle and Ariane Szafarz. Controlling firms through the majority voting rule. *Physica A: Statistical Mechanics and its Applications*, 355(2-4):509–529, 2005.
11. Peter Christen. *Data Matching - Concepts and Techniques for Record Linkage, Entity Resolution, and Duplicate Detection*. Data-centric systems and applications. Springer, 2012.
12. Caroline Fohlin. Trends in business organization: Do participation and cooperation increase competitiveness? edited by horst siebert. tubingen: J. c. b. mohr, 1995. pp. viii, 292. dm 108. *The Journal of Economic History*, 58(1):292–293, 1998.
13. James B Glattfelder. *Ownership networks and corporate control: mapping economic power in a globalized world*. PhD thesis, ETH Zurich, 2010.
14. Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, pages 855–864. ACM, 2016.
15. David Liben-Nowell and Jon M. Kleinberg. The link prediction problem for social networks. In *CIKM*, pages 556–559. ACM, 2003.
16. Tiago Pimentel, Adriano Veloso, and Nivio Ziviani. Fast node embeddings: Learning ego-centric representations. In *ICLR (Workshop)*. OpenReview.net, 2018.
17. Andrea Romei, Salvatore Ruggieri, and Franco Turini. The layered structure of company share networks. In *2015 IEEE International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–10. IEEE, 2015.