DAPER joint learning from partially structured Graph Databases

Marwa El Abri, Philippe Leray, Nadia Essoussi

▶ **To cite this version:**

Marwa El Abri, Philippe Leray, Nadia Essoussi. DAPER joint learning from partially structured Graph Databases. Third annual International Conference on Digital Economy (ICDEc 2018), 2018, Brest, France. pp.129-138, ⟨10.1007/978-3-319-97749-2_10⟩. ⟨hal-01804057⟩

HAL Id: hal-01804057

https://hal.science/hal-01804057v1

Submitted on 9 Apr 2020

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

HAL Authorization

DAPER joint learning from partially structured Graph Databases

Marwa El abri[1,2], Philippe Leray[2], and Nadia Essoussi[1]

[1] LARODEC Laboratory ISG, Université de Tunis, Tunisia.
[2] LS2N UMR CNRS 6004, DUKe research group, University of Nantes, France
marwa.el-abri@etu.univ-nantes.fr , philippe.leray@univ-nantes.fr
nadia.essoussi@isg.rnu.tn

Abstract. In this paper, we are interested in learning specific probabilistic relational models, named Directed Acyclic Probabilistic Entity Relationship (DAPER) models, from partially structured databases. Algorithms for such a learning task already exist for structured data coming from a relational database. They have been also extended to partially structured data stored in a graph database where the Entity Relationship (ER) schema is first identified from data, and then the DAPER dependency structure is learnt for this specific ER schema. We propose in this work a joint learning from partially structured graph databases where we want to learn at the same time the ER schema and the probabilistic dependencies. The Markov Logic Network (MLN) formalism is an efficient solution for this task. We show with an illustrative example that MLN structure learning can effectively learn both parts of the DAPER model in one single task, with a comparative precision, but with a very high complexity.

Keywords: Probabilistic Relational Model, Directed Acyclic Probabilistic Entity Relationship Model, Markov Logic Network, Graph Database, Partially Structured Data, Structure Learning.

1 Introduction

Statistical Relational Learning (SRL) [1, 2] combines the descriptive power of relational modeling with the flexibility of statistical learning to develop models and learning algorithms capable of representing complex relationships among entities in uncertain domains. Several models emerged, with Probabilistic Relational Models such as Direct Acyclic Probabilistic Entity Relationship models (DAPER) [3] or Relational Bayesian Networks (RBN) [4, 5], but also frameworks such as Markov Logic Networks (MLN) [6], ProbLog [7] or Bayesian Logic Programs (BLP) [8] where the relational information is described using First-Order Logic.

In this paper, we are interested in learning DAPER models from partially structured databases. Algorithms for such a learning task already exist for structured data coming from a relational database [9–14]. It has been also extended to partially structured data stored in a graph database [15] where the Entity Relationship (ER) schema is first identified from data, and then the DAPER dependency structure is learnt for this specific ER schema.

We propose in this work a joint learning from partially structured graph databases, where we want to learn at the same time the ER schema and the probabilistic dependencies. The MLN formalism is an efficient solution for this task. The ER schema and the set of probabilistic dependencies of the models are described in the same logical way, and dedicated structure learning algorithm could be able to retrieve both information at the same time. The logical formulas used by MLN formalism can also manage exceptions, i.e. data which are not coherent with an underlying structured model, as we have to deal with when working with partially structured data.

This paper is organized as follows. Section 2 is dedicated to define the DAPER model and the MLN formalism. Section 3 describes how a DAPER can be expressed with Markov Logical framework. In Section 4, we describe our proposed method. A detailed illustrative example is presented in Section 5. Some conclusions and future works are drawn in Section 6.

2 Background

Direct Acyclic Probabilistic Entity Relationship models (DAPER) [3] and Markov Logic Networks (MLN) [6] are Probabilistic Relational Models that can be learnt from uncertain and relational information. We briefly present both models in the section before describing how they can be related in the next section.

2.1 DAPER model

A DAPER [3] is a probabilistic extension of Bayesian network [16] based on the representation of Entity-Relationship (ER) model defined by [17]. DAPER model is composed by a set of entity classes $\mathcal{E}$, relationship classes $\mathcal{R}$ and attribute classes $\mathcal{A}(X)$ (with $X \in (\mathcal{E} \cup \mathcal{R})$) where all attributes $\mathcal{A}(X)$ are random variables that can depend on each other. Generally, the probabilistic dependency structure is graphically defined by a set of parents $pa(X.A)$ for each attribute object $X.A$, associated to a local distribution corresponding to this set of variables. In [18], these local distributions are defined for each attribute $X.A \in \mathcal{A}(\mathcal{E} \cup \mathcal{R})$ by the conditional probability distribution denoted $P(X.A|pa(X.A))$. Figure 1 shows an example of a DAPER model for the university domain where a student's grade for one course depends both on the student's intelligence and on the difficulty

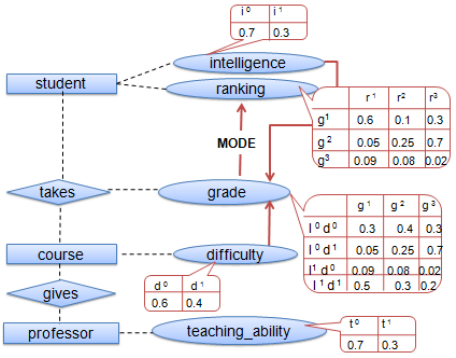

Fig. 1. Example of a DAPER model (inspired from [5]).

of the course. In Figure 1, the probabilistic dependencies between attributes are represented by red solid arcs.

Parents $pa(X.A)$ of a given attribute can be defined by using paths (also named slot chains) in the ER model [9, 5] or more general constraints [3]. Formally, a slot chain is a set of slots $\rho_1, \rho_2, ..., \rho_n$ where ρ is a (inverse) reference slot which relates objects of a class $X \in (\mathcal{E} \cup \mathcal{R})$ to objects of a class Y. *takes.student* is the student associated to the corresponding registration. $student.student^{-1}$ is an inverse reference slot corresponding to all the registrations of a given student. As an example of slot chain, $student.student^{-1}.course$ will correspond to all the courses taken by a particular student.

Another important concept in DAPER is the notion of aggregator that comes into play when there is dependency between the objects that have one-to-many or many-to-many relations. In Figure 1, *student.ranking* depends on $student.student^{-1}.grade$. As a student can take more than one course, *student.ranking* will depend on grades of more than one *takes* object and this number will not be the same for all students. So, in order to get a summary of such dependencies, aggregators (such as MODE or MIN) are introduced.

The structure learning task for DAPER models aims at identifying the probabilistic dependencies between attributes (and the corresponding conditional probability distributions) given an ER model and its instantiation. DAPER models are used to be learnt from relational database for which the Entity Relationship (ER) schema is already defined and the data are well structured. Thus, all existing structure learning approaches are based on this ER schema for learning the set of probabilistic dependencies from structured data [9], [10], [11, 12] and [13, 14]. [15] proposed to learn both the ER schema and the set of probabilistic dependencies from partially structured databases where the ER schema is not a priori defined. In this paper, we aim to learn both ER schema and the graph dependencies at the same time from partially structured data.

2.2 Markov Logic Networks

When DAPER can be seen as relational extensions of Bayesian networks, Markov Logic Networks (MLN) [19, 6] are probabilistic relational models that generalize both full first-order logic and Markov networks. Formally, a Markov Logic Network is defined as a set of pairs (F_i, φ_i), where F_i is a formula in first-order logic and φ_i is a weight associated with this formula.

As an example, let us consider a university domain with four predicates $Smart(x), Easy(y), Take(x, y)$ and $Grade(x, y, val)$. Equation 1 shows a possible (and very simple) MLN composed of one formula for this domain.

$$F : \forall x, y \;\; Grade(x, y, val) \implies Smart(x) \wedge Easy(y) \wedge Take(x, y) \qquad (1)$$

Some algorithms have been proposed for MLN structure learning, i.e. identifying the set of formulas, or more usually their clausal form, and the corresponding weights, from data [19–21].

3 Expressing a DAPER in Markov Logic

Based on [22], we describe in this section how to define a DAPER in the Markov Logic framework. We first define $predicates_{\mathcal{E}}$, the set of predicates corresponding to the set of entity classes $\mathcal{E}$ of the ER schema, where each entity class E will correspond to a $predicate_E(object)$ describing if this object belongs to this class. In the same way, we define $predicates_{\mathcal{R}}$, the set of binary predicates corresponding to the set of $\mathcal{R}$ in the ER schema, where each relationship class R corresponds to a $predicate_R(object1, object2)$. We finally define $predicates\_\mathcal{A}$ which are divided into two classes $predicates\_\mathcal{A}(\mathcal{E})$ and $predicates\_\mathcal{A}(\mathcal{R})$. Each $predicate_A(object, value)$ corresponds to an attribute class $A \in \mathcal{A}(\mathcal{E})$ where the first argument is the entity class to which this attribute is associated and the second one is a value for this attribute. Each $predicate_A(object1, object2, value)$ corresponds to an attribute class $A \in \mathcal{A}(\mathcal{R})$ with the first two arguments are the entities involved in the corresponding relationship class.

In the MLN formalism, we also have to declare that a separate weight must be learned for each formula obtained by grounding that variable to one of its values. Finally, some prior knowledge can be added. For instance, in our case, each object belongs to a unique class.

Let us illustrate these steps by considering the DAPER described in Figure 1. The corresponding MLN will contain the knowledge base described in Table 1 which defines the set of predicates and the prior knowledge.

After defining all these concepts, we can now define the ER schema as a set of clauses $\mathcal{F}_{ER}$, one for each relationship, $f_{ER} : predicate_R \Rightarrow predicate_{E1} \wedge predicate_{E2}$ when the relationship R is defined for entities $E1 \times E2$. With perfectly structured data, these formulas are certain.

Predicates ER	**Predicates Attributes**	**Prior knowledge**
student(s)	s_intelligence(s,ival!)	student(x) ⇔ !course(x)
course(c)	s_ranking(s,rval!)	student(x) ⇔ !professor(x)
professor(p)	c_difficulty(c,dval!)	course(x) ⇔! professor(x)
takes(s,c)	p_teaching_ability(p,taval!)	
gives(p,c)	t_grade(s,c,gval!)	

Table 1. Knowledge base corresponding to the DAPER described in Figure 1.

Equation 2 shows us the exact formulas corresponding to the ER schema defined in Figure 1.

$$\begin{aligned} takes(s,c) &\Rightarrow student(s) \wedge course(c) \\ gives(p,c) &\Rightarrow professor(p) \wedge course(c) \end{aligned} \tag{2}$$

We can finally describe the probabilistic dependencies of the DAPER with a set of formulas $\mathcal{F}_{PD}$. Each formula is defined with one head which is related to one predicate attribute (and a specific value) and whose body involves the values of the parents of this attribute (as defined in the DAPER dependency structure) and the logical description of the slot chain between this attribute and its parents. For instance, in the DAPER described in Figure 1, parents of *takes.grade* are *takes.student.intelligence* and *takes.course.difficulty*. The logical description of this dependency involves 12 formulas (as the total number of values in the conditional probability distribution) with the pattern described in equation 3.

$$\begin{aligned} takes\_grade(x,y,valG) \Rightarrow student(x) \wedge s\_intellligence(x,valI) \\ \wedge\ course(y) \wedge c\_difficulty(y,valD) \end{aligned} \tag{3}$$

4 DAPER joint learning using MLN framework

We propose in this work a joint DAPER learning from partially structured graph databases, where we want to learn at the same time the ER schema and the probabilistic dependencies by using Markov Logic Network formalism. As described in Section 3, both parts of the model (ER schema and probabilistic dependencies) can be described in the same logical way in Markov logic framework. MLN structure learning algorithm could then be able to retrieve simultaneously both information.

4.1 DAPER joint learning

In order to learn a DAPER from partially structure data, we propose to learn first an MLN from the same database, and then to extract (1) the ER schema and (2) the set of probabilistic by extracting the associated formulas in the MLN.

Identification of the ER schema. The first format of formulas we are looking for is the set of formulas $\mathcal{F}_{ER}$ that describe the ER schema, as described in Section 3, $f_{ER} : predicate_R \Rightarrow predicate_{E1} \wedge predicate_{E2}$, or in a clausal form $\neg R \vee E_1 \vee E_2$.

As we are dealing with partially structured data, several formulas can be extracted for the same relationship. We propose to consider only the strongest f_{ER} in term of weight (in normalized absolute value) as the relevant relationship. We then apply a user-defined threshold $\lambda_{er} \in [0, 1]$ to identify our ER schema.

Identification of the probabilistic dependencies. The second format of clauses we are looking for is the set of formulas $\mathcal{F}_{PD}$ that correspond to the probabilistic dependencies. f_{PD}: $\text{predicate}_Y \Rightarrow \text{K}_1.\text{predicate}_{X1} \wedge,...,\wedge\, \text{K}_n.\text{predicate}_{Xn}$. As mentioned in the Equation 3, a probabilistic dependency between one random variable and its parents can generate several logical formulas, one for each possible configuration of the variables. We then propose to extract such a dependency only when the average of the normalized absolute values of the weights $avg\varphi$ of the corresponding compatible f_{PD} is greater than another user-defined threshold $\lambda_{pd} \in [0, 1]$.

As the formulas are described in their clausal form, it's not always possible to identify the direction of the dependency, as defined in DAPER framework, nor the possible aggregation function. Only the slot chains between variables can be retrieved from the formulas. We then summarize our dependency graph with an undirected graph where the nodes are the attributes, the edges are the probabilistic dependencies discovered, labelled with their associated slot chains.

4.2 Evaluation process

Several metrics have been proposed to evaluate DAPER structure learning algorithms. Concerning the ER schema, as described in [15], we use a Hamming distance (RSHD_ER) between the graphs describing the original ER schema and the learnt one in order to evaluate the quality of this step. The task concerning the probabilistic dependencies is more complex. We cannot use directly existing metrics described for instance in [23, 14] to compare the original probabilistic directed model and the undirected one obtained in the Section 4.1. Thus, we propose to use a weighted Hamming distance between the undirected counterpart of the original model and the one created from the logic formulas of the MLN (RSHD_PD). [14] proposed a same idea for directed models, with a weight for a given edge defined as the similarity between the slot chains associated to this

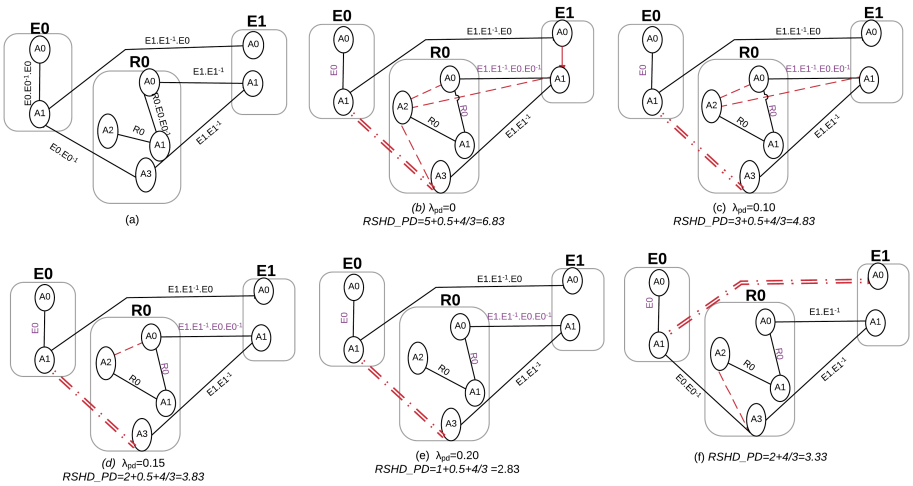

Fig. 2. Undirected dependency structure (a) of the theoretical DAPER, (b,c,d,e) extracted from the MLN with various thresholds λ_{pd} and (f) from the concurrent solution proposed in [15], with RSHD_PD between these models and (a). Information in solid lines correspond to true discovered edges compared with model (a). Information in dashed lines and double dash-dot lines correspond respectively to additional and missing edges.

edge in both models. This computation does not take into account the starting class of the slot chain, which leads to overestimating the error when comparing an empty slot chain to a non empty one. We propose here to simply add this starting class in the similarity computation in order to solve this problem.

5 Illustrative example

In this section, we introduce an illustrative example detailing all the steps of our approach.

5.1 Experimental protocol

We have used the sampling process defined by [24] to first generate a theoretical DAPER with 2 entity classes and 1 relationship class, 8 attributes (2 or 3 per class) and 7 probabilistic dependencies with slot chain length between 0 and 2) and then to sample a relational database instance from this DAPER, with 3000 instances.

As performed in [15], this relational database is transformed into a graph databases. Some "exceptions" (with respect to the relational schema) have been

added by transforming some existing relationship signatures by another ones not conform with the underlying ER model. The final graph database is finally containing partially structured data where we control the percentage of exceptions (30% in the following experiments). This partially structured graph database is also converted into a knowledge database with 7681 ground predicates.

We used two different approaches to learn our DAPER model. The first one consists in using an MLN learning algorithm from the given knowledge base (Beam search from Alchemy software[1], with the following parameters: Maximum predicates per clause = 6, Penalization of weighted pseudo-likelihood = 0.1) and then extract the ER schema and the undirected dependency structure as described in Section 4.1. The second approach consists in using the method proposed by [15] where the ER schema and the dependency structure are learnt separately from the graph database, with an identification threshold $\lambda = .5$ for ER identification and a maximum possible slot chain length $k_{max} = 2$ for the dependency structure identification. Experiments have been carried out on a dedicated PC with Intel(R) Core(TM)i7-4600M CPU 2.7GHz, 64 bits architecture, 8 Gb RAM memory and under Windows 7.

5.2 Results and evaluation

Once the MLN is learnt from the previous database, we automatically extract the following formulas.

$$\begin{aligned} \varphi &= 4.707 : relationshipclass0 \Rightarrow entityclass2 \wedge entityclass1 \\ \varphi &= 0.334 : relationshipclass0 \Rightarrow entityclass1 \wedge entityclass2 \end{aligned} \tag{4}$$

The first formula with the higher weight corresponds to the original ER schema, when the one with the lower weight corresponds to the exceptions present in our partially structured database. When applying a threshold $\lambda_{er} = 0.5$ to extract the ER schema from the formulas, we are able to perfectly identify it, so RSHD_ER=0 in this experiment. We also extract 66 clauses $\mathcal{F}_{\mathcal{PD}}$ corresponding to probabilistic dependencies that are converted into an undirected structure.

Figure 2(a) provides the undirected dependency structure of the theoretical DAPER. Figure 2(b) to (d) respectively provide the structure derived from the MLN formulas with a threshold $\lambda_{pd} = 0, 0.10, 0.15$ and 0.20. As we can see the solution we propose here is able to perfectly retrieve the underlying ER schema and to identify the dependency structure with an increasing quality when λ_{pd} increases and an optimal RSHD_PD=2.83.

The result of the concurrent algorithm is described in Figure 2(f). It also perfectly retrieves the underlying ER schema and identifies the dependency structure with RSHD_PD=3.33. Also, we have compared both approaches in term

[1] https://alchemy.cs.washington.edu/

of running time. We have spent more than 10 hours by using our MLN learning based algorithm, but only 2 minutes from the initial concurrent method.

6 Conclusion and future works

In this paper, we were interested in learning Directed Acyclic Probabilistic Entity Relationship (DAPER) models, from partially structured databases. We proposed here a joint learning from partially structured graph databases, where we simultaneously learn the ER schema and the probabilistic dependencies. The Markov Logic Network formalism appeared as an efficient solution for this task. We show with an illustrative example that MLN structure learning can effectively identify both part of the DAPER model in one single task, with a comparative precision, but with a very higher complexity. MLN semantics, and more specifically formulas described in clausal form, also restrict ourself by only identify undirected dependency structures, whereas DAPER are directed probabilistic models. In an opposite way, the logical formulas used by MLN to describe both the ER schema and the probabilistic dependencies can manage exceptions, i.e. data which are not coherent with an underlying structured model, which is not possible with the DAPER framework.

As our objective is obtaining an efficient probabilistic framework dealing with partially structured graph databases, we are now interested by improving this work into several directions. We are currently working on more complete experiments to test this method with all datasets used in [15] to consolidate our results. Since MLN structure learning algorithms seem to suffer from complexity issues, we are also interested by other probabilistic and relational frameworks derived from Logic such as ProbLog models [7]. If we can confirm that DAPER structure learning is really less complex than its MLN/ProbLog counterparts, a last perspective would be to improve MLN/ProbLog structure learning by first learning a more restrictive by less complex DAPER model.

References

1. B. Taskar, P. Abbeel, M.-F. Wong, and D. Koller, "Relational markov networks," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds. MIT Press, 2007.
2. J. Neville and D. Jensen, "Relational dependency networks," in *Introduction to Statistical Relational Learning*, L. Getoor and B. Taskar, Eds., 2005.
3. D. Heckerman and M. Meek, "Probabilistic entity-relationship models, PRMs, and plate models," *ICML*, pp. 55–60, 2004.
4. D. Koller and A. Pfeffer, "Probabilistic frame-based systems." *AAAI/IAAI*, pp. 580–587, 1998.

5. L. Getoor, "Learning statistical models from relational data," Ph.D. dissertation, Stanford, 2001.
6. M. Richardson and P. Domingos, "Markov Logic Networks," *Machine Learning*, vol. 62, no. 1-2, pp. 107–136, 2006.
7. L. Raedt, A. Kimmig, and H. Toivonen, "Problog: A probabilistic prolog and its application in link discovery." in *IJCAI*, M. M. Veloso, Ed., 2007, pp. 2462–2467.
8. K. Kersting and L. Raedt, *Bayesian Logic Programming: Theory and Tool*. MIT Press, 2007, ch. 10.
9. N. Friedman, L. Getoor, D. Koller, and A. Pfeffer, "Learning probabilistic relational models," *In IJCAI*, pp. 1300–1309, 1999.
10. N. Ettouzi, P. Leray, and M. Ben Messaoud, "An exact approach to learning probabilistic relational model," in *PGM*, 2016, pp. 171–182.
11. M. Maier, K. Marazopoulou, and D. Jensen, "Reasoning about independence in probabilistic models of relational data," *CoRR*, vol. abs/1302.4381, 2013.
12. S. Lee and V. Honavar, "On learning causal models from relational data." in *AAAI*, 2016, pp. 3263–3270.
13. L. Xiao-Lin and H. Xiang-Dong, "A hybrid particle swarm optimization method for structure learning of probabilistic relational models," *Information Sciences*, vol. 283, pp. 258 – 266, 2014, new Trend of Computational Intelligence in Human-Robot Interaction.
14. M. Ben Ishak, "Probabilistic relational models: learning and evaluation," Ph.D. dissertation, Université de Nantes, Université de Tunis, Jun. 2015.
15. M. El Abri, P. Leray, and N. Essoussi, "Daper learning from (partially structured) graph database," in *IEEE AICCSA*, 2017.
16. J. Pearl, *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. San Francisco, CA, USA: Morgan Kaufmann Publishers Inc., 1988.
17. P.-S. Chen, "The entity-relationship model: Toward a unified view of data," *ACM Trans. Database Syst.*, vol. 1, no. 1, pp. 9–36, Mar. 1976.
18. F. Kaelin and D. Precup, "A study of approximate inference in probabilistic relational models." *JMLR.org*, vol. 13, pp. 315–330, 2010.
19. S. Kok and P. Domingos, "Learning structure of Markov Logic Networks," in *ICML*, 2005, pp. 441–448.
20. P. Domingos and S. Kok, "Learning Markov Logic Networks structure via hypergraph lifting," in *ICML*. ACM, 2009, pp. 505–512.
21. S. Kok and P. Domingos, "Learning Markov Logic Networks using structural motifs," in *ICML*, 2010, pp. 551–558.
22. P. Domingos and M. Richardson, "Markov logic: A unifying framework for statistical relational learning," in *Proceeding of the ICML-2004 workshop on statistical relational learning and its connections to other fields*, 2004, pp. 49–54.
23. M. Maier, K. Marazopoulou, D. Arbour, and D. Jensen, "A sound and complete algorithm for learning causal models from relational data," *CoRR*, vol. abs/1309.6843, 2013.
24. M. Ben Ishak, P. Leray, and N. Ben Amor, "Probabilistic relational model benchmark generation," *Intell. Data Anal*, pp. 615–635, 2016.