CERIAS Tech Report 2007-83
Relational Dependency Networks
by Jennifer Neville and David Jensen
Center for Education and Research
Information Assurance and Security
Purdue University, West Lafayette, IN 47907-2086

1 Relational Dependency Networks

Jennifer Neville and David Jensen
Computer Science Department
University of Massachusetts Amherst
Amherst, MA 01003-4610, USA
{jneville | jensen}@cs.umass.edu
http://kdl.cs.umass.edu

Recent work on graphical models for relational data has demonstrated significant improvements in classification and inference when models represent the dependencies among instances. Despite its use in conventional statistical models, the assumption of instance independence is contradicted by most relational datasets. For example, in citation data there are dependencies among the topics of a paper's references, and in genomic data there are dependencies among the functions of interacting proteins. In this chapter we present relational dependency networks (RDNs), a graphical model that is capable of expressing and reasoning with such dependencies in a relational setting. We discuss RDNs in the context of relational Bayes networks and relational Markov networks and outline the relative strengths of RDNs—namely, the ability to represent cyclic dependencies, simple methods for parameter estimation, and efficient structure learning techniques. The strengths of RDNs are due to the use of *pseudolikelihood* learning techniques, which estimate an efficient approximation of the full joint distribution. We present learned RDNs for a number of real-world datasets and evaluate the models in a prediction context, showing that RDNs identify and exploit cyclic relational dependencies to achieve significant performance gains over conventional conditional models.

1.1 Introduction

Many datasets routinely captured by businesses and organizations are relational in nature, yet until recently most machine learning research has focused on "flattened" propositional data. Instances in propositional data record the characteristics of homogeneous and statistically independent objects; instances in relational data record the characteristics of heterogeneous objects and the relations among those objects. Examples of relational data include citation graphs, the World Wide Web, genomic structures, fraud detection data, epidemiology data, and data on interrelated people, places, and events extracted from text documents.

The presence of *autocorrelation* provides a strong motivation for using relational techniques for learning and inference. Autocorrelation is a statistical dependency between the values of the same variable on related entities and is a nearly ubiquitous characteristic of relational datasets [Jensen and Neville, 2002]. More formally, autocorrelation is defined with respect to a set of related instance pairs $P_R = \{(o_i, o_j) : o_i, o_j \in O\}$; it is the correlation between the values of a variable X on the instance pairs $(o_i.x, o_j.x)$ such that $(o_i, o_j) \in P_R$. Recent analyses of relational datasets have reported autocorrelation in the following variables:

- Topics of hyperlinked web pages [Chakrabarti et al., 1998, Taskar et al., 2002]
- Industry categorization of corporations that share boards members [Neville and Jensen, 2000]
- Fraud status of cellular customers who call common numbers [Cortes et al., 2001]
- Topics of coreferent scientific papers [Taskar et al., 2001, Neville and Jensen, 2003]
- Functions of proteins located together in a cell [Neville and Jensen, 2002]
- Box-office receipts of movies made by the same studio [Jensen and Neville, 2002]
- Industry categorization of corporations that co-occur in new stories [Bernstein et al., 2003]
- Tuberculosis infection among people in close contact [Getoor et al., 2001]

When relational data exhibit autocorrelation there is a unique opportunity to improve model performance because inferences about one object can inform inferences about related objects. Indeed, recent work in relational domains has shown that *collective inference* over an entire dataset results in more accurate predictions than conditional inference for each instance independently [e.g., Chakrabarti et al., 1998, Neville and Jensen, 2000, Lu and Getoor, 2003], and that the gains over conditional models increase as autocorrelation increases [Jensen et al., 2004].

Joint relational models are able to exploit autocorrelation by estimating a joint probability distribution over an entire relational dataset and collectively inferring the labels of related instances. Recent research has produced several novel types of graphical models for estimating joint probability distributions for relational data that consist of non-independent and heterogeneous instances [e.g., Getoor et al., 2001, Taskar et al., 2002]. We will refer to these models as *probabilistic relational models* (PRMs).[1] PRMs extend traditional graphical models such as Bayesian

1. Several previous papers [e.g., Friedman et al., 1999, Getoor et al., 2001] use the term

networks to relational domains, removing the assumption of independent and identically distributed instances that underlies conventional learning techniques. PRMs have been successfully evaluated in several domains, including the World Wide Web, genomic data, and scientific literature.

Directed PRMs, such as relational Bayes networks[2] (RBNs) [Getoor et al., 2001], can model autocorrelation dependencies if they are structured in a manner that respects the acyclicity constraint of the model. While domain knowledge can sometimes be used to structure the autocorrelation in an acyclic manner, often an acyclic ordering is unknown or does not exist. For example, in genetic pedigree analysis there is autocorrelation among the genes of relatives [Lauritzen and Sheehan, 2003]. In this domain, the casual relationship is from ancestor to descendent so we can use the temporal parent-child relationship to structure the dependencies in an acyclic manner (i.e., parents' genes will never be influenced by the genes of their children). However, given a set of hyperlinked web pages, there is little information to use to determine the causal direction of the dependency between their topics. In this case, we can only represent an (undirected) correlation between the topics of two pages, not a (directed) causal relationship. The acyclicity constraint of directed PRMs precludes the learning of arbitrary autocorrelation dependencies and thus severely limits the applicability of these models in relational domains.

Undirected PRMs, such as relational Markov networks (RMNs) [Taskar et al., 2002], can represent and reason with arbitrary forms of autocorrelation. However, research for these models has focused primarily on parameter estimation and inference procedures. The current RMN learning algorithm does not select features—model structure must be pre-specified by the user. While in principle it is possible for RMN techniques to learn cyclic autocorrelation dependencies, inefficient parameter estimation makes this difficult in practice. Because parameter estimation requires multiple rounds of inference over the entire dataset, it is impractical to incorporate it as a subcomponent of feature selection. Recent work on conditional random fields for sequence analysis includes a feature selection algorithm [McCallum, 2003] that could be extended for RMNs. However, the algorithm abandons estimation of the full joint distribution and uses pseudolikelihood estimation, which makes the approach tractable but removes some of the advantages of reasoning with the full joint distribution.

In this chapter, we outline relational dependency networks (RDNs), an extension of dependency networks [Heckerman et al., 2000] for relational data. RDNs can

---

probabilistic relational model to refer to a specific model that is now often called a *relational Bayesian network* [Koller, personal communication]. In this paper, we use PRM in its more recent and general sense.

2. We use the term relational Bayesian network to refer to Bayesian networks that have been upgraded to model relational databases. The term has also been used by Jaeger [1997] to refer to Bayesian networks where the nodes correspond to relations and their values represent possible interpretations of those relations in a specific domain.

represent and reason with the cyclic dependencies required to express and exploit autocorrelation during collective inference. In this regard, they share certain advantages of RMNs and other undirected models of relational data [Chakrabarti et al., 1998, Domingos and Richardson, 2001]. Also, to our knowledge, RDNs are the first PRM capable of *learning* cyclic autocorrelation dependencies. RDNs offer a relatively simple method for structure learning and parameter estimation, which results in models that are easier to understand and interpret. In this regard they share certain advantages of RBNs and other directed models [Sanghai et al., 2003, Heckerman et al., 2004]. The primary distinction between RDNs and other existing PRMs is that RDNs are an approximate model. RDN models approximate the full joint distribution and thus are not guaranteed to specify a coherent probability distribution. However, the quality of the approximation will be determined by the data available for learning—if the models are learned from large datasets, and combined with Monte Carlo inference techniques, the approximation should not be a disadvantage.

We start by reviewing the details of dependency networks for propositional data. Then we describe the general characteristics of PRM models and outline the specifics of RDN learning and inference procedures. We evaluate RDN learning and inference algorithms on both synthetic and real-world datasets, presenting learned RDNs for subjective evaluation and evaluating the models in a prediction context. Of particular note, all the real-world datasets exhibit multiple autocorrelation dependencies that were automatically discovered by the RDN learning algorithm. Finally, we review related work and conclude with a discussion of future directions.

1.2 Dependency Networks

Graphical models represent a joint distribution over a set of variables. The primary distinction between Bayesian networks, Markov networks, and dependency networks (DNs) is that dependency networks are an approximate representation. DNs approximate the joint distribution with a set of conditional probability distributions (CPDs) that are learned independently. This approach to learning results in significant efficiency gains over exact models. However, because the CPDs are learned independently, DN models are not guaranteed to specify a *consistent* joint distribution. This precludes DNs from being used to infer causal relationships and limits the applicability of exact inference techniques. Nevertheless, DNs can encode predictive relationships (i.e., dependence and independence) and Gibbs sampling inference techniques [e.g., Neal, 1993] can be used to recover a full joint distribution, regardless of the consistency of the local CPDs.

1.2.1 DN Representation

Dependency networks are an alternative form of graphical model that approximate the full joint distribution with a set of conditional probability distributions that are each learned independently. A DN encodes probabilistic relationships among a set of variables $\mathbf{X}$ in a manner that combines characteristics of both undirected and directed graphical models. Dependencies among variables are represented with a bidirected graph $G = (V, E)$, where conditional independence is interpreted using graph separation, as with undirected models. However, as with directed models, dependencies are quantified with a set of conditional probability distributions P. Each node $v_i \in V$ corresponds to an $X_i \in \mathbf{X}$ and is associated with a probability distribution conditioned on the other variables, $P(v_i) = p(x_i|\mathbf{x} - \{x_i\})$. The parents of node i are the set of variables that render X_i conditionally independent of the other variables ($p(x_i|pa_i) = p(x_i|\mathbf{x} - \{x_i\})$), and G contains a directed edge from each parent node v_j to each child node v_i ($e(v_j, v_i) \in E$ iff $X_j \in pa_i$). The CPDs in P do not necessarily factor the joint distribution so we cannot compute the joint probability for a set of values $\mathbf{x}$ directly. However, given G and P, a joint distribution can be recovered through Gibbs sampling (see below for details). From the joint distribution, we can extract any probabilities of interest.

1.2.2 DN Learning

Both the structure and parameters of DN models are determined through learning the local CPDs. The DN learning algorithm learns a separate distribution for each variable X_i, conditioned on the other variables in the data (i.e., $\mathbf{X} - \{X_i\}$). Any conditional learner can be used for this task (e.g., logistic regression, decision trees). The CPD is included in the model as $P(v_i)$ and the variables selected by the conditional learner form the parents of X_i (e.g., if $p(x_i|\{\mathbf{x} - x_i\}) = \alpha x_j + \beta x_k$ then $pa_i = \{x_j, x_k\}$). The parents are then reflected in the edges of G appropriately. If the conditional learner is not selective (i.e., the algorithm does not select a subset of the features), the DN model will be fully connected (i.e., $pa_i = \mathbf{x} - \{x_i\}$). In order to build understandable DNs, it is desirable to use a selective learner that will learn CPDs that use a subset of the variables.

1.2.3 DN Inference

Although the DN approach to structure learning is simple and efficient, it can result in an inconsistent network, both structurally and numerically. In other words, there may be no joint distribution from which each of the CPDs can be obtained using the rules of probability. Learning the CPDs independently with a selective conditional learner can result in a network that contains a directed edge from X_i to X_j, but not from X_j to X_i. This is a structural inconsistency—X_i and X_j are dependent but X_j is not represented in the CPD for X_i. In addition, learning the CPDs independently from finite samples may result in numerical inconsistencies in parameter estimates, where the derived joint distribution does not sum to one. In practice, Heckerman et al. [2000] show that DNs are nearly consistent if learned from large datasets because the data serve a coordinating function to ensure some degree of consistency among the CPDs. However, even when a DN is inconsistent, approximate inference techniques can still be used to estimate a full joint distribution and extract probabilities of interest. Gibbs sampling can be used to recover a full joint distribution, regardless of the consistency of the local CPDs, provided that each X_i is discrete and its CPD is positive [Heckerman et al., 2000].

1.3 Relational Dependency Networks

Several characteristics of DNs are particularly desirable for modeling relational data. First, learning a collection of conditional models offers significant efficiency gains over learning a full joint model. This is generally true, but is even more pertinent to relational settings where the feature space is very large. Second, networks that are easy to interpret and understand aid analysts' assessment of the utility of the relational information. Third, the ability to represent cycles in a network facilitates reasoning with autocorrelation, a common characteristic of relational data. In addition, whereas the need for approximate inference is a disadvantage of DNs for propositional data, due to the complexity of relational model graphs in practice, all PRMs use approximate inference.

Relational dependency networks extend DNs to work with relational data in much the same way that RBNs extend Bayesian networks and RMNs extend Markov networks. These extensions take a graphical model formalism and *upgrade* [Kersting, 2003] it to a first-order logic representation with an entity-relationship model. We start by describing the general characteristics of probabilistic relational models and then discuss the details of RDNs in this context.

1.3.1 Probabilistic Relational Models

PRMs represent a joint probability distribution over the attributes of a relational dataset. When modeling propositional data with a graphical model, there is a single graph G that that comprises the model. In contrast, there are three graphs associated with models of relational data: the *data graph* G_D, the *model graph* G_M, and the *inference graph* G_I. These correspond to the *skeleton*, *model*, and *ground graph* as outlined in Heckerman et al. [2004].

First, the relational dataset is represented as a typed, attributed data graph $G_D = (V_D, E_D)$. For example, consider the data graph in Figure 1.1a. The nodes V_D represent objects in the data (e.g., authors, papers) and the edges E_D represent relations among the objects (e.g., author-of, cites).[3] Each node $v_i \in V_D$ and edge $e_j \in E_D$ is associated with a type $T(v_i) = t_{v_i}$ (e.g., paper, cited-by). Each item[4] type $t \in T$ has a number of associated attributes $\mathbf{X^t} = (X_1^t, ..., X_m^t)$ (e.g., topic,

3. We use rectangles to represent objects, circles to represent random variables, dashed lines to represent relations, and solid lines to represent probabilistic dependencies.
4. We use the generic term "item" to refer to objects or links.

year). Consequently, each object v_i and link e_j is associated with a set of attribute values determined by their type $\mathbf{X_{v_i}^{t_{v_i}}} = (X_{v_i1}^{t_{v_i}}, ..., X_{v_im}^{t_{v_i}})$, $\mathbf{X_{e_j}^{t_{e_j}}} = (X_{e_j1}^{t_{e_j}}, ..., X_{e_jm'}^{t_{e_j}})$. A PRM model represents a joint distribution over the values of the attributes in the data graph, $\mathbf{x} = \{\mathbf{x_{v_i}^{t_{v_i}}} : v_i \in V, t_{v_i} = T(v_i)\} \cup \{\mathbf{x_{e_j}^{t_{e_j}}} : e_j \in E, t_{e_j} = T(e_j)\}$.

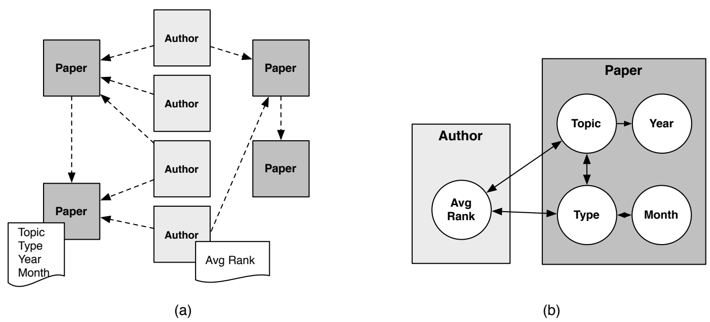

Figure 1.1 Example (a) data graph and (b) model graph.

Next, the dependencies among attributes are represented in the model graph $G_M = (V_M, E_M)$. Attributes of an item can depend probabilistically on other attributes of the same item, as well as on attributes of other related objects or links in G_D. For example, the topic of a paper may be influenced by attributes of the authors that wrote the paper. Instead of defining the dependency structure over attributes of specific objects, PRMs define a generic dependency structure at the level of item types. Each node $v \in V_M$ corresponds to an X_k^t, where $t \in T \wedge X_k^t \in \mathbf{X^t}$. The set of attributes $\mathbf{X_k^t} = (X_{ik}^t : (v_i \in V \vee e_i \in E) \wedge T(i) = t)$ is tied together and modeled as a single variable. This approach of typing items and tying parameters across items of the same type is an essential component of PRM learning. It enables generalization from a *single* instance (i.e., one data graph) by decomposing the data graph into *multiple* examples of each item type (e.g., all paper objects), and building a joint model of dependencies between and among attributes of each type.

As in conventional graphical models, each node is associated with a probability distribution conditioned on the other variables. Parents of X_k^t are either: (1) other attributes associated with type t_k (e.g., paper *topic* depends on paper *type*), or (2) attributes associated with items of type t_j where items t_j are related to items t_k in G_D (e.g., paper *topic* depends on author *rank*). For the latter type of dependency, if the relation between t_k and t_j is one-to-many, the parent consists of a set of attribute values (e.g., author ranks). In this situation, current PRM models use aggregation functions to generalize across heterogeneous items (e.g., one paper may have two authors while another may have five). Aggregation functions are used to either map sets of values into single values, or to combine a set of probability distributions into a single distribution.

Consider the RDN model graph G_M in Figure 1.1b. It models the data in Figure 1.1a, which has two object types: paper and author. In G_M, each item type is represented by a plate, and each attribute of each item type is represented as a node. Edges characterize the dependencies among the attributes at the type level. The representation uses a modified plate notation—dependencies among attributes of the same object are contained inside the rectangle and arcs that cross the boundary of the rectangle represent dependencies among attributes of related objects. For example, $month_i$ depends on $type_i$, while $avgrank_j$ depends on the $type_k$ and $topic_k$ for all papers k related to author j in G_D.

There is a nearly limitless range of dependencies that could be considered by algorithms learning PRM models. In propositional data, learners model a fixed set of attributes intrinsic to each object. In contrast, in relational data, learners must decide how much to model (i.e., how much of the relational neighborhood around an item can influence the probability distribution of a item's attributes). For example, a paper's topic may depend of the topics of other papers written by its authors—but what about the topics of the references in those papers or the topics of other papers written by coauthors of those papers? Two common approaches to limiting search in the space of relational dependencies are: (1) exhaustive search of all dependencies within a fixed-distance neighborhood (e.g., attributes of items up to k links away), or (2) greedy iterative-deepening search, expanding the search in the neighborhood in directions where the dependencies improve the likelihood.

Finally, during inference, a PRM uses a model graph G_M and a data graph G_D to instantiate an inference graph $G_I = (V_I, V_E)$ in a process sometimes called "rollout." The rollout procedure used by PRMs to produce G_I is nearly identical to the process used to instantiate sequence models such as hidden Markov models. G_I represents the probabilistic dependencies among all the variables in a single test set (here G_D is usually different from G'_D used for training). The structure of G_I is determined by both G_D and G_M—each item-attribute pair in G_D gets a separate, local copy of the appropriate CPD from G_M. The relations in G_D constrain the way that G_M is rolled out to form G_I. PRMs can produce inference graphs with wide variation in overall and local structure because the structure of G_I is determined by the specific data graph, which typically has non-uniform structure. For example, Figure 1.2 shows the RDN from Figure 1.1b rolled out over a dataset of three authors and three papers, where P_1 is authored by A_1 and A_2, P_2 is authored by A_2 and A_3, and P_3 is authored by A_3. Notice that there are a variable number of authors per paper. This illustrates why current PRMs use aggregation in their CPDs—for example, the CPD for paper-type must be able to deal with a variable number of author ranks.

1.3.2 RDN Representation

Relational dependency networks encode probabilistic relationships in a similar manner to DNs, extending the representation to a relational setting. RDNs use

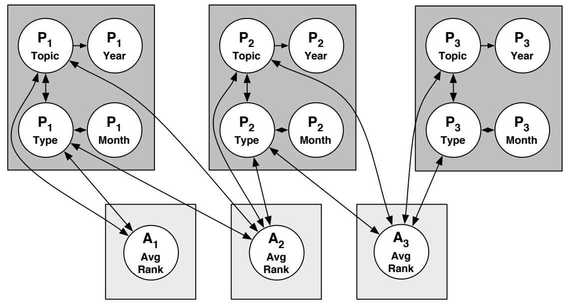

Figure 1.2 Example PRM inference graph.

a bidirected model graph G_M with a set of conditional probability distributions P. Each node $v_i \in V_M$ corresponds to an $X_k^t \in \mathbf{X^t}$, $t \in T$ and is associated with a conditional distribution $p(x_k^t \,|\, pa_{x_k^t})$. Figure 1.1b illustrates an example RDN model graph for the data graph in Figure 1.1a. The graphical representation illustrates the qualitative component (G_D) of the RDN—it does not depict the quantitative component (P) of the model, which consists of CPDs that use aggregation functions. Although conditional independence is infered using an undirected view of the graph, bidirected edges are useful for representing the set of variables in each CPD. For example, in Figure 1.1b the CPD for *year* contains *topic* but the CPD for *topic* does not contain *type*. This depicts any inconsistencies that result from the RDN learning technique.

1.3.3 RDN Learning

Learning a PRM model consists of two tasks: learning the dependency structure among the attributes of each object type, and estimating the parameters of the local probability models for an attribute given its parents. Relatively efficient techniques exist for learning both the structure and parameters of RBN models. However, these techniques exploit the requirement that the CPDs *factor* the full distribution—a requirement that imposes acyclicity constraints on the model and precludes the learning of arbitrary autocorrelation dependencies. On the other hand, although in principle it is possible for RMN techniques to learn cyclic autocorrelation dependencies, inefficiencies due to calculating the normalizing constant Z in undirected models make this difficult in practice. Calculation of Z requires a summation over all possible states $\mathbf{X}$. When modeling the joint distribution of propositional data, the number of states is exponential in the number of attributes (i.e., $O(2^m)$). When modeling the joint distribution of relational data, the number of states is exponential in the number of attributes and *the number of instances*. If there are N objects, each with m attributes, then the total number of states is $O(2^{Nm})$. For any reasonable-size dataset, a single calculation of Z is an enormous computational burden. Feature selection generally requires repeated parameter estimation while measuring the change in likelihood affected by each attribute, which would require recalculation of Z on each iteration.

The RDN learning algorithm uses a more efficient alternative—estimating the set of conditional distributions independently rather than jointly. This approach is based on *pseudolikehood* techniques [Besag, 1975], which were developed for modeling spatial datasets with similar autocorrelation dependencies. Pseudolikelihood estimation avoids the complexities of estimating Z and the requirement of acyclicity. In addition, this approach can utilize existing techniques for learning conditional probability distributions of relational data such as first-order Bayesian classifiers [Flach and Lachiche, 1999], structural logistic regression [Popescul et al., 2003], or ACORA [Perlich and Provost, 2003].

Instead of optimizing the log-likelihood of the full joint distribution, we optimize the pseudo-loglikelihood for each variable independently, conditioned on all other attribute values in the data:

$$PL(G_D;\theta) = \sum_{t \in T} \sum_{X_i^t \in X^t} \sum_{v \in T(v)} p(x_{vi}^t | pa_{x_{vi}^t}) \tag{1.1}$$

With this approach we give up the asymptotic efficiency guarantees of maximum likelihood estimators. However, under some general conditions the consistency of maximum pseudolikelihood estimators can be established [Geman and Graffine, 1987], which implies that, as sample size $\rightarrow \infty$, pseudolikelihood estimators will produce unbiased estimates of the true parameters.

On the surface 1.1 may appear similar to the joint distribution specified by an RBN. However, the CPDs in the pseudolikelihood are not required to factor the joint distribution of G_D. More specifically, when we consider the variable X_{vi}^t, we condition on the values of the parents $pa_{X_{vi}^t}$ regardless of whether the estimation of $pa_{X_{vi}^t}$ was conditioned on X_{vi}^t. The parents of X_{vi}^t may include the values of other attributes (e.g., $X_{vi'}^{t'}$ such that $t' \neq t$ or $i' \neq i$) or the values of the same variable on related items (e.g., $X_{v'i}^t$ such that $v' \neq v$).

The RDN learning algorithm is similar to the DN learning algorithm, except we use a relational probability estimation algorithm to learn a set of conditional models, maximizing the pseudolikelihood for each variable separately. The algorithm input consists of:

- G_D: a relational data graph
- R: a conditional relational learner
- $\mathbf{Q^t}$: a set of queries that specify the types T and limits the relational neighborhood that is considered in R for each T
- $\mathbf{X^t}$: a set of attributes for each item type

Table 1.1 outlines the learning algorithm in pseudocode. It cycles over each attribute of each item type and learns a separate CPD, conditioned on the other values in

Table 1.1 RDN Learning Algorithm

Learn RDN $(G_D, R, \mathbf{Q^t}, \mathbf{X^t})$:
$P \leftarrow \emptyset$
For each $t \in T$:
    For each $X_k^t \in \mathbf{X^t}$:
        Use R to learn a CPD for X_k^t given the attributes $\{X^t_{k' \neq k}\} \cup \mathbf{X^{t' \neq t}}$
         in the relational neighborhood defined by Q^t.
        $P \leftarrow P \cup CPD_{X_k^t}$
Use P to form G_M.

---

the training data. We discuss details of the subcomponents (querying and relational learners) next.

1.3.3.1 Queries

The queries specify the relational neighborhoods that will be considered by the conditional learner R, and their structure defines a typing over instances in the database. Subgraphs are extracted from a larger graph database using the visual query language QGraph [Blau et al., 2001]. Queries allow for variation in the number and types of objects and links that form the subgraphs and return collections of all matching subgraphs from the database.

For example, consider the query in Figure 1.3a.[5] The query specifies match criteria for a target item (paper) and its local relational neighborhood (authors and references). The example query matches all research papers that were published in 1995 and returns for each paper a subgraph that includes all authors and references associated with the paper. Figure 1.3b shows a hypothetical match to this query: a paper with two authors and seven references.

The query defines a typing over the objects of the database (e.g., people that have authored a paper are categorized as *authors*) and specifies the relevant relational context for the target item type in the model. For example, given this query the model R would model the distribution of a paper's attributes given the attributes of the paper itself and the attributes of its related authors and references. The queries are a means of restricting model search. Instead of setting a depth limit on the extent of the search, the analyst has a more flexible means with which to limit the search (e.g., we can consider other papers written by the paper's authors but not other authors of the paper's references).

---

5. We have modified the QGraph representation to conform to our convention of using rectangles to represent objects and dashed lines to represent relations.

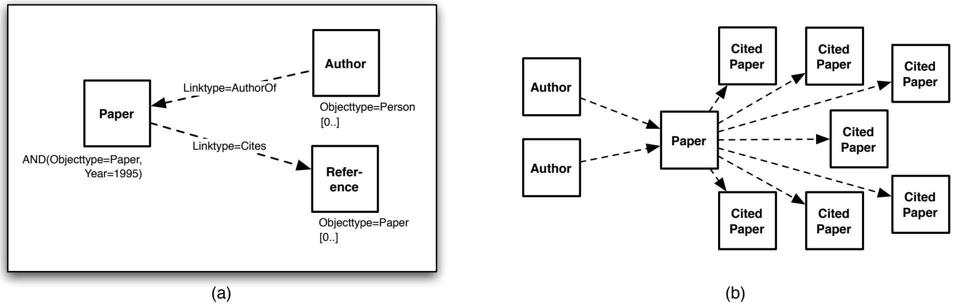

Figure 1.3 (a) Example QGraph query: Textual annotations specify match conditions on attribute values; numerical annotations (e.g., [0..]) specify constraints on the cardinality of matched objects (e.g., zero or more authors), and (b) matching subgraph.

1.3.3.2 Conditional Relational Learners

The conditional relational learner R is used for both parameter estimation and structure learning in RDNs. The variables selected by R are reflected in the edges of G appropriately. If R selects all of the available attributes, the RDN model will be fully connected.

In principle, any conditional relational learner can be used as a subcomponent to learn the individual CPDs. In this paper, we discuss the use of two different conditional models—relational Bayesian classifiers (RBCs) [Neville et al., 2003b] and relational probability trees (RPTs) [Neville et al., 2003a].

Relational Bayesian Classifiers
RBCs extend Bayesian classifiers to a relational setting. RBC models treat heterogeneous relational subgraphs as a homogenous set of attribute multisets. For example, when considering the references of a single paper the publication dates of those references form multisets of varying size (e.g., {1995, 1995, 1996}, {1975, 1986, 1998, 1998}). The RBC assumes each value of a multiset is independently drawn from the same multinomial distribution.[6] This approach is designed to mirror the independence assumption of the naive Bayesian classifier. In addition to the conventional assumption of attribute independence, the RBC also assumes attribute value independence within each multiset.

For a given item type T, the query scope specifies the set of item types $\mathbf{T_R}$ that form the relevant relational neighborhood for T. For example, in Figure 1.3a $T = paper$ and $\mathbf{T_R} = \{paper, author, reference, authorof, cites\}$. To estimate the CPD for

6. Alternative constructions are possible but prior work [Neville et al., 2003b] has shown this approach achieves superior performance over a wide range of conditions.

attribute X on items T (e.g., paper topic), the model considers all the attributes associated with the types in $\mathbf{T_R}$. RBCs are non-selective models so all the attributes are included as parents:

$$p(x|pa_x) \propto \prod_{t \in \mathbf{T_R}} \prod_{X_i^t \in X^t} \prod_{v \in T_R(x)} p(x_{vi}^t|x)\ p(x)$$

Relational Probability Trees

RPTs are selective models that extend classification trees to a relational setting. RPT models also treat heterogeneous relational subgraphs as a set of attribute multisets, but instead of modeling the multisets as independent values drawn from a multinomial, the RPT algorithm uses aggregation functions to map a set of values into a single feature value. For example, when considering the publication dates on references of a research paper the RPT could construct a feature that tests whether the *average* publication date was after 1995. Figure 1.4 provides an example RPT learned on citation data.

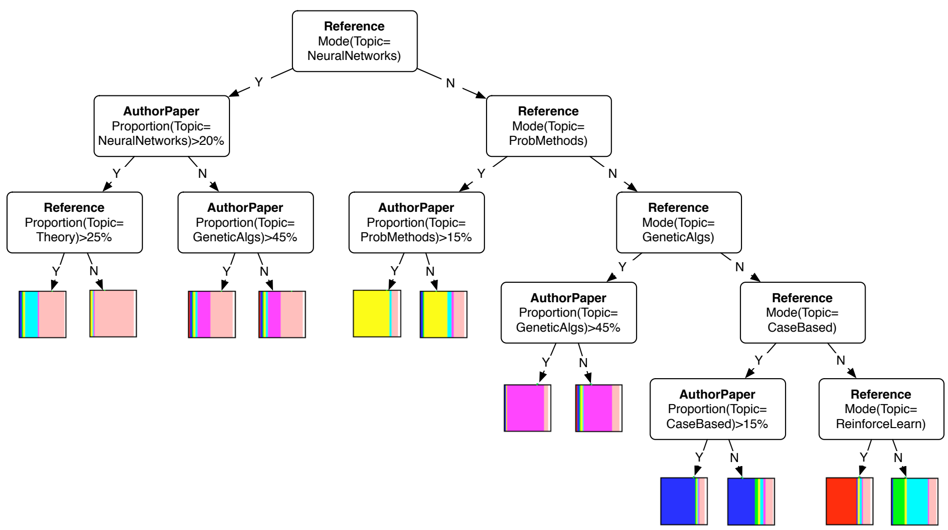

Figure 1.4 Example RPT to predict machine-learning paper topic.

The RPT algorithm automatically constructs and searches over aggregated relational features to model the distribution of the target variable X. The algorithm constructs features from the attributes associated with the types specified in the query. The algorithm considers four classes of aggregation functions to group multiset values: *Mode, Count, Proportion, Degree*. For discrete attributes, the algorithm constructs features for all unique values of an attribute. For continuous attributes, the algorithm constructs features for a number of different discretizations, binning the values by frequency (e.g., *year* > 1992). Count, proportion, and degree features consider a number of different thresholds (e.g., $proportion(A) > 10\%$). Feature scores are calculated using chi-square to measure correlation between the feature and the class. The algorithm uses pre-pruning in the form of a p-value cutoff and a depth cutoff to limit tree size. All experiments reported herein used $\alpha = 0.05/|attributes|$, *depth cutoff*=7, and considered 10 thresholds and discretizations per feature.

The RPT learning algorithm adjusts for biases towards particular features due to degree disparity and autocorrelation in relational data [Jensen and Neville, 2002, 2003]. We have shown that RPTs build significantly smaller trees than other conditional models and achieve equivalent, or better, performance [Neville et al., 2003a]. These characteristics of RPTs are crucial for learning understandable RDN models and have a direct impact on inference efficiency because smaller trees limit the size of the final inference graph.

1.3.4 RDN Inference

The RDN inference graph G_I is potentially much larger than the original data graph. To model the full joint distribution there must be a separate node (and CPD) for each attribute value in G_D. To construct G_I, the set of template CPDs in P is rolled out over the test-set data graph. Each item-attribute pair gets a separate, local copy of the appropriate CPD. Consequently, the total number of nodes in the inference graph will be $\sum_{v \in V_D} |\mathbf{X}^{\mathbf{T(v)}}| + \sum_{e \in E_D} |\mathbf{X}^{\mathbf{T(e)}}|$. Rollout facilitates generalization across data graphs of varying size—we can learn the CPD templates from one data graph and apply the model to a second data graph with a different number of objects by rolling out more CPD copies. This approach is analogous to other graphical models that tie distributions across the network and rollout copies of model templates (e.g., hidden Markov models).

We use Gibbs sampling for inference in RDN models. Gibbs sampling can be used to extract a unique joint distribution, regardless of the consistency of the model [Heckerman et al., 2000].

Table 1.2 outlines the inference algorithm. To estimate a joint distribution, we start by rolling out the model G_M onto the target dataset G_D, forming the inference graph G_I. The values of all unobserved variables are initialized to values drawn from their prior distributions. Gibbs sampling then iteratively relabels each unobserved variable by drawing from its local conditional distribution, given the current state of the rest of the graph. After a sufficient number of iterations (*burn in*), the values will be drawn from a stationary distribution and we can use the samples to estimate probabilities of interest.

For prediction tasks we are often interested in the marginal probabilities associated with a single variable X (e.g., paper topic). Although Gibbs sampling may be a relatively inefficient approach to estimating the probability associated with a joint

Table 1.2 RDN Inference Algorithm

Infer RDN $(G_D, G_M, P, iter, burnin)$:
$G_I(V_I, E_I) \leftarrow (\emptyset, \emptyset)$ \\\\ *form* G_I *from* G_D *and* G_M
For each $t \in T$ *in* G_M:
  For each $X_k^t \in \mathbf{X^t}$ *in* G_M:
    For each $v_i \in V_D$ *s.t.* $T(v_i) = t$:
      $V_I \leftarrow V_I \cup \{X_{v_i k}^t\}$
      For each $v_j \in V_D$ *s.t.* $X_{v_j} \in pa_{X_{v_i k}^t}$:
        $E_I \leftarrow E_I \cup \{e_{ij}\}$
For each $v \in V_I$: \\\\ *initialize Gibbs sampling*
  Randomly initialize x_v to an arbitrary value
$S \leftarrow \emptyset$ \\\\ *Gibbs sampling procedure*
For $i \in iter$:
  For each $v \in V_I$, in random order:
    Resample x'_v from $p(x_v | \mathbf{x} - \{x_v\})$
    $x_v \leftarrow x'_v$
    If $i > burnin$:
      $S \leftarrow S \cup \{\mathbf{x}\}$
Use samples S to estimate probabilities of interest

---

assignment of values of X (e.g., when $|X|$ is large), it is often reasonably fast to estimate the marginal probabilities for each X.

There are many implementation issues that can improve the estimates obtained from a Gibbs sampling chain, such as length of burn-in and number of samples. For the experiments reported in this paper we used fixed-length chains of 2000 samples (each iteration re-labels every value sequentially) with burn-in set at 100. Empirical inspection indicated that the majority of chains had converged by 500 samples.

1.4 Experiments

The experiments in this section demonstrate the utility of RDNs as a joint model of relational data. First, we use synthetic data to assess the impact of training-set size and autocorrelation on RDN learning and inference, showing that accurate models can be learned at reasonable dataset sizes and that the model is robust to varying levels of autocorrelation. Next, we learn RDN models of three real-world datasets to illustrate the types of domain knowledge that the models discover automatically. In addition, we evaluate RDN models in a prediction context, where only a single attribute is unobserved in the test set, and report significant performance gains compared to two conditional models.

1.4.1 Synthetic Data Experiments

To explore the effects of training-set size and autocorrelation on RDN learning and inference, we generated homogeneous data graphs with autocorrelation due to an underlying (hidden) group structure. Each object has four boolean attributes: X_1, X_2, X_3 and X_4. The data generation procedure uses a simple RDN where X_1 is autocorrelated (through objects one link away), X_2 depends on X_1, and the other two attribute have no dependencies. To generate data with autocorrelated X_1 values, we used manually specified conditional models for $p(X_1|\mathbf{X_1}_R, X_2)$.

We compare two different RDN models: RDN_{RBC} uses RBCs for the component model R; RDN_{RPT} uses RPT for R. The RPT performs feature selection, which may result in structural inconsistencies in the learned RDN. The RBC does not use feature selection so any deviation from the true model is due to numerical inconsistencies alone. Note that the two models do not consider identical feature spaces so we can only roughly assess the impact of feature selection by comparing RDN_{RBC} and RDN_{RPT} results.

1.4.1.1 RDN Learning

The first set of synthetic experiments examines the effectiveness of the RDN learning algorithm. Theoretical analysis indicates that, in the limit, the true parameters will maximize the pseudolikelihood function. This indicates that the pseudolikelihood function, evaluated at the learned parameters, will be no greater than the pseudolikelihood of the true model (on average). To evaluate the quality of the RDN parameter estimates, we calculated the pseudolikelihood of the test-set data using both the true model (used to generate the data) and the learned models. If the pseudolikelihood given the learned parameters approaches the pseudolikelihood given the true parameters, then we can conclude that parameter estimation is successful. We also measured the standard error of the pseudolikelihood estimate for a single test-set using learned models from 10 different training sets. This illustrates the amount of variance due to parameter estimation.

Figure 1.5 graphs the pseudo-loglikelihood of learned models as a function of training-set size for three levels of autocorrelation. Training-set size was varied at the levels $\{50, 100, 250, 500, 1000, 5000\}$. We varied $p(X_1|\mathbf{X_1}_R, X_2)$ to generate data with approximate levels of autocorrelation corresponding to $\{0.25, 0.50, 0.75\}$. At each training set size (and autocorrelation level), we generated 10 test sets. For each test set, we generated 10 training sets and learned RDNs. Using each learned model, we measured the pseudolikelihood of the test set (size 250) and averaged the results over the 10 models.

Figure 1.5 plots the mean pseudolikelihood of the test sets for both the learned models and the RDN used for data generation, which we refer to as *True Model*. The top row reports experiments with data generated from an RDN_{RPT}, where we

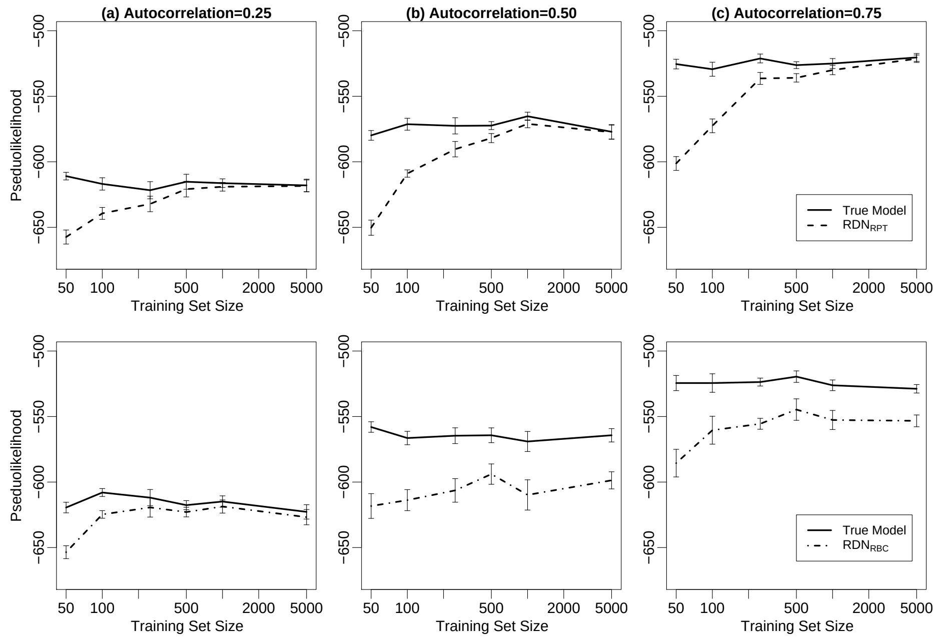

Figure 1.5 Evaluation of RDN learning.

learned RDN_{RPT} models. The bottom row reports experiments with data generated from an RDN_{RBC}, where we learned RDN_{RBC} models.

These experiments show that the learned RDN_{RPT} models are a good approximation to the true model by the time training-set size reaches 500, and that RDN learning is robust with respect to varying levels of autocorrelation. As expected, however, when training-set size is small, the RDNs are a better approximation for datasets with low levels of autocorrelation (see Figure 1.5a).

There appears to be little difference between the RDN_{RPT} and RDN_{RBC} when autocorrelation is low, but otherwise the RDN_{RBC} needs significantly more data to estimate the parameters accurately. This may be in part due to the model's lack of selectivity, which necessitates the estimation of a greater number of parameters. However, there is little improvement even when we increase the size of the training sets to 10,000 objects. Furthermore, the discrepancy between the estimated model and the true model is greatest when autocorrelation is moderate. This indicates that the inaccuracies may be due to the naive Bayes independence assumption and its tendency to produce biased probability estimates [Zadrozny and Elkan, 2001].

1.4.1.2 RDN Inference

The second set of synthetic experiments evaluates the RDN inference procedure in a prediction context, where only a single attribute is unobserved in the test set. We generated data in the manner described above and learned RDNs for X_1. At each autocorrelation level, we generated 10 training sets (size 500) and learned RDNs. For each training set, we generated 10 test sets (size 250) and used the learned models to infer marginal probabilities for the class labels of the test set instances. To evaluate the predictions, we report area under the ROC curve (AUC).[7] These experiments used the same levels of autocorrelation outlined above.

We compare the performance of three types of models. First, we measure the performance of RPT and RBC models. These are *conditional models* that reason about each instance independently and do not use the class labels of related instances. Next, we measure the performance of the two RDN models described above: RDN_{RBC} and RDN_{RPT}. These are *collective models* that reason about instances jointly, using the inferences about related instances to improve overall performance. Lastly, we measure performance of the two RDN models while allowing the true labels of related instances to be used during inference. This demonstrates the level of performance possible if the RDNs could infer the true labels of related instances with perfect accuracy. We refer to these as *ceiling models*: RDN_{RBC}^{ceil} and RDN_{RPT}^{ceil}.

Note that conditional models can reason about autocorrelation dependencies in a limited manner by using the attributes of related instances. For example, if there is a correlation between the words on a webpage and its topic, and the topics of hyperlinked webpages are autocorrelated, then we can improve the inference about a single page by modeling the contents of its neighboring pages. Recent work has shown that collective models are a low-variance means of reducing bias that work by modeling the autocorrelation dependencies directly [Jensen et al., 2004]. Conditional models are also able to exploit autocorrelation dependencies through modeling the attributes of related instances, but variance increases dramatically as the number of attributes increases.

During inference we varied the number of known class labels in the test set, measuring performance on the remaining unlabeled instances. This serves to illustrate model performance as the amount of information seeding the inference process increases. We expect performance to be similar when other information seeds the inference process—for example, when some labels can be inferred from intrinsic attributes, or when weak predictions about many related instances serve to constrain the system. Figure 1.6 graphs AUC results for each of the models as the level of known class labels is varied.

In all configurations, RDN_{RPT} performance is equivalent, or better than, RPT performance. This indicates that even modest levels of autocorrelation can be exploited to improve predictions using RDN_{RPT} models. RDN_{RPT} performance is indistinguishable from that of RDN_{RPT}^{ceil} except when autocorrelation is high and there are no labels to seed inference. In this situation, there is little information to

7. Squared-loss results are qualitatively similar to the AUC results reported in Figure 1.6.

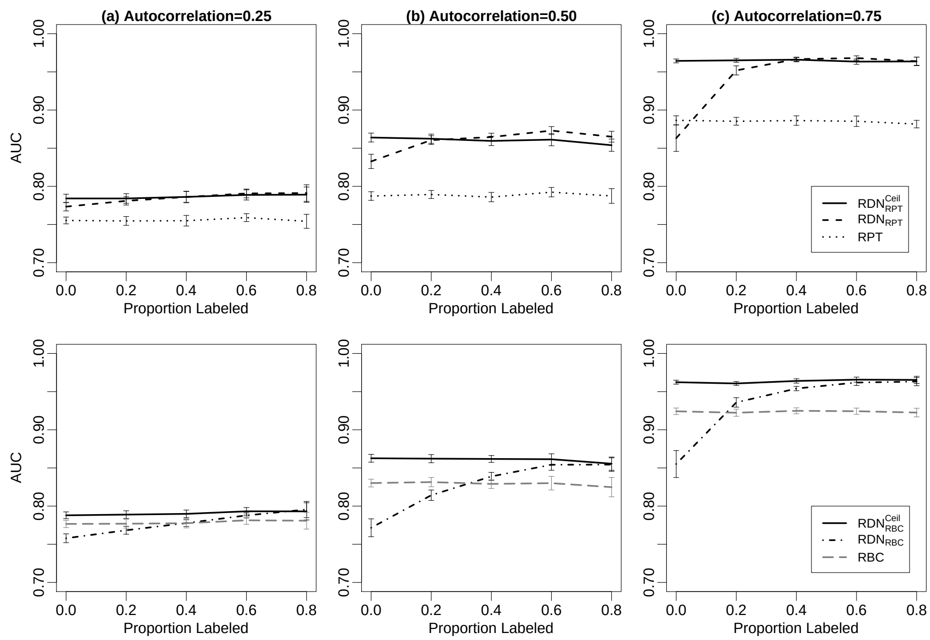

Figure 1.6 Evaluation of RDN inference.

constrain the system during inference so the model cannot fully exploit the autocorrelation dependencies. When there is no information to anchor the predictions, there will be an identifiability problem—symmetric labelings that are highly autocorrelated, but with opposite values, will be equally likely. In situations where there is little seed information, identifiability problems can bias RDN performance towards random.

In contrast, RDN_{RBC} performance is superior to RBC performance only when there is moderate to high autocorrelation and sufficient seed information. When autocorrelation is low, the RBC model is comparable to both the RDN^{ceil}_{RBC} and RDN_{RBC} models. Even when autocorrelation is moderate or high, RBC performance is still relatively high. Since the RBC model is low-variance and there are only four attributes in our datasets, it is not surprising that the RBC model is able to exploit autocorrelation to improve performance. What is more surprising is that RDN_{RBC} requires substantially more seed information than RDN_{RPT} in order to reach ceiling performance. This indicates that our choice of model should take test-set characteristics (e.g., number of known labels) into consideration.

1.4.2 Empirical Data Experiments

We learned RDN models for three real-world relational datasets to illustrate the types of domain knowledge that can be garnered, and evaluated the models in a prediction context, where the values of a single attribute are unobserved. Figure 1.7 depicts the objects and relations in each dataset.

The first dataset is drawn from the Internet Movie Database (IMDb: www.imdb.com). We collected a sample of 1,382 movies released in the United States between 1996 and 2001, with their associated actors, directors, and studios. In total, this sample contains approximately 42,000 objects and 61,000 links.

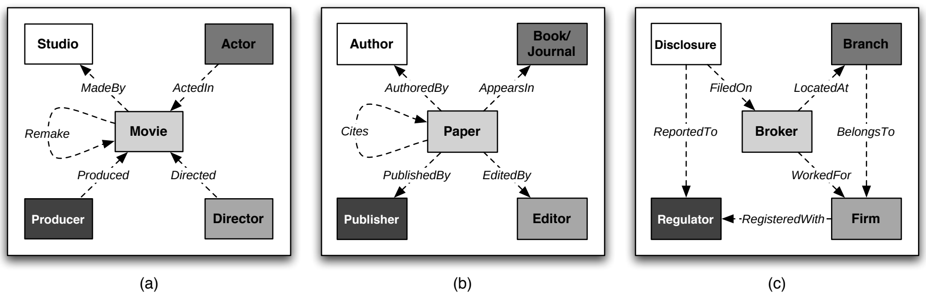

Figure 1.7 Data schemas for (a) IMDb, (b) Cora, (c) NASD.

The second dataset is drawn from Cora, a database of computer science research papers extracted automatically from the web using machine learning techniques [McCallum et al., 1999]. We selected the set of 4,330 machine-learning papers along with associated authors, cited papers, and journals. The resulting collection contains approximately 13,000 objects and 26,000 links. For classification, we sampled the 1669 papers published between 1993 and 1998.

The third dataset is from the National Association of Securities Dealers (NASD) [Neville et al., 2005]. It is drawn from NASD's Central Registration Depository (CRD©) system, which contains data on approximately 3.4 million securities brokers, 360,000 branches, 25,000 firms, and 550,000 disclosure events. Disclosures record disciplinary information on brokers, including information on civil judicial actions, customer complaints, and termination actions. Our analysis was restricted to small and moderate-size firms with fewer than 15 brokers, each of whom has an approved NASD registration. We selected a set of 10,000 brokers who were active in the years 1997-2001, along with 12,000 associated branches, firms, and disclosures.

1.4.2.1 RDN Models

The RDN models in Figures 1.8-1.10 continue with the RDN representation introduced in Figure 1.1b. Each item type is represented by a separate plate. Arcs inside a plate represent dependencies among the attributes of a single object, and arcs crossing the boundaries of plates represent dependencies among attributes of related objects. An arc from x to y indicates the presence of one or more features

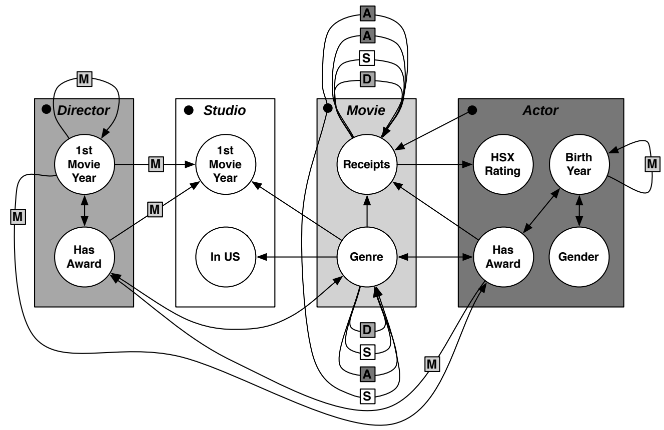

Figure 1.8 Internet Movie database RDN.

of x in the conditional model learned for y. When the dependency is on attributes of objects more than a single link away, the arc is labeled with a small rectangle to indicate the intervening related-object type. For example, in Figure 1.8 movie genre is influenced by the genres of other movies made by the movie's director, so the arc is labeled with a small D rectangle.

In addition to dependencies among attribute values, relational learners may also learn dependencies between the structure of relations (edges in G_D) and attribute values. *Degree* relationships are represented by a small black circle in the corner of each plate—arcs from this circle indicate a dependency between the number of related objects and an attribute value of an object. For example, in Figure 1.8 movie receipts are influenced by the number of actors in the movie.

For each dataset, we learned RDNs using queries that include all neighbors up to two links away in the data graph. For example in the IMDb, when learning a model of movie attributes we considered the attributes of associated actors, directors, producers and studios, as well as movies related to those objects.

On the IMDb data, we learned an RDN model for ten discrete attributes including actor gender and movie opening weekend receipts (*>$2million*). Figure 1.8 shows the resulting RDN model. Four of the attributes—movie receipts, movie genre, actor birth year, and director first movie year—exhibit autocorrelation dependencies. Exploiting this type of dependency has been shown to significantly improve classification accuracy of RMNs compared to RBNs, which cannot model cyclic dependencies [Taskar et al., 2002]. However, to exploit autocorrelation, RMNs must be instantiated with the appropriate clique templates—to date there is no RMN algorithm for *learning* autocorrelation dependencies. RDNs are the first PRM capable of learning cyclic autocorrelation dependencies.

On the Cora data, we learned an RDN model for seven attributes including paper

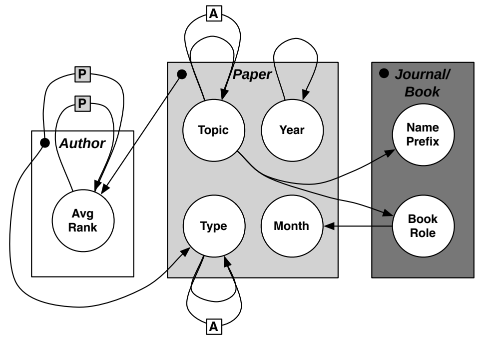

Figure 1.9 Cora machine-learning papers RDN.

topic (e.g., neural networks) and journal name prefix (e.g., IEEE). Figure 1.9 shows the resulting RDN model. Again we see that four of the attributes exhibit autocorrelation. Note that when a dependency is on attributes of objects a single link away, the arc is unlabeled. For example, the unlabeled self-loops from paper variables indicates dependencies on the same variables in cited papers. In particular, the topic of a paper depends not only on the topics of other papers that it cites, but also on the topics of other papers written by the authors. This model is a good reflection of our domain knowledge about machine learning papers.

On the NASD data, we learned an RDN model for eleven attributes including broker is-problem and disclosure type (e.g., customer complaint). Figure 1.10 shows the resulting RDN model. Again we see that four of the attributes exhibit autocorrelation. Subjective inspection by NASD analysts indicates that the RDN has automatically uncovered statistical relationships that confirm the intuition of domain experts. These include temporal autocorrelation of risk (past problems are indicators of future problems) and relational autocorrelation of risk among brokers at the same branch—indeed, fraud and malfeasance are usually social phenomena, communicated and encouraged by the presence of other individuals who also wish to commit fraud [Cortes et al., 2001]. Importantly, this evaluation was facilitated by the interpretability of the RDN model—experts are more likely to trust, and make regular use of, models they can understand.

1.4.2.2 Prediction

We evaluated the learned models on prediction tasks in order to assess (1) whether autocorrelation dependencies among instances can be used to improve model accuracy, and (2) whether the RDN models, using Gibbs sampling, can effectively infer labels for a network of instances. To do this, we compared the same three classes of models used in Section 1.4.1: RPTs and RBCs, RDNs, and ceiling RDNs.

Figure 1.11 shows AUC results for each of the models on the three prediction tasks. Figure 1.11a graphs the results of the RDN_{RPT} models, compared to the

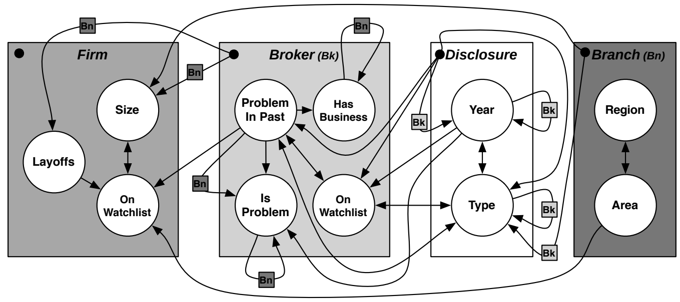

Figure 1.10 RDN for NASD data for 1999.

RPT conditional model. Figure 1.11b graphs the results of the RDN_{RBC} models, compared to the *RBC* conditional model. We used the following prediction tasks: movie receipts for IMDb, paper topic for Cora, and broker is-problem for NASD.

The graphs show AUC for the most prevalent class, averaged over a number of training/test splits. We used temporal samples where we learned models on one year of data and applied the model to the subsequent year. We used two-tailed, paired t-tests to assess the significance of the AUC results obtained from the trials. The t-tests compare the RDN results to each of the other two models with a null hypothesis of no difference in the AUC.

When using the RPT as the conditional learner (Figure 1.11a), RDN performance is superior to RPT performance on all tasks. The difference is statistically significant for two of the three tasks. This indicates that autocorrelation is both present in the data and identified by the RDN models. The RPT can sometimes use

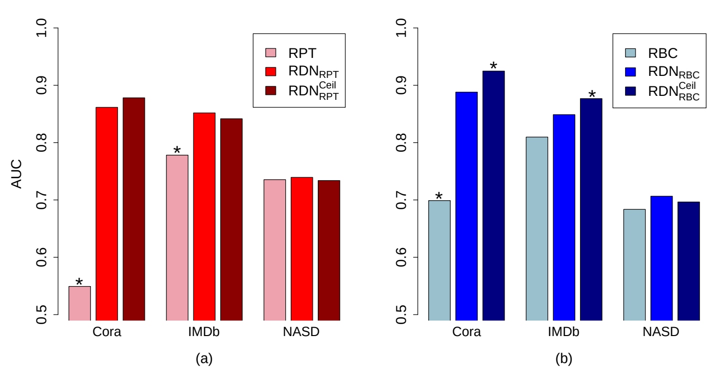

Figure 1.11 AUC results for (a) RDN_{RPT} and RPT models, and (b) RDN_{RBC} and RBC models. Asterisks denote model performance that is significantly different ($p < 0.10$) from RDN_{RPT} and RDN_{RBC}.

attributes of related items to effectively represent and reason with autocorrelation dependencies. However, in some cases the attributes other than the class label contain little information about the class labels of related instances. This is the case for Cora—RPT performance is close to random because no other attributes influence paper topic (see Figure 1.9). On all tasks, the RDN models achieve comparable performance to the ceiling models. This indicates that the RDN model achieved the same level of performance as if it had access to the true labels of related objects. On the NASD data, the RDN performance is slightly higher than that of the ceiling model. We note, however, that the ceiling model only represents a probabilistic ceiling—the RDN may perform better if an incorrect prediction for one object improves inferences about related objects.

Similarly, when using the RBC as the conditional learner (Figure 1.11b), the performance of RDN models is superior to the RBC models on all tasks and statistically significant for two of the tasks. However, the RDN models achieve comparable performance to the ceiling models on only one of the tasks. This may be another indication that RDN models combined with a non-selective conditional learner (e.g., RBCs) will experience increased variance during the Gibbs sampling process, and thus they may need more seed information during inference to achieve the near-ceiling performance. We should note that although the RDN_{RBC} models do not significantly outperform the RDN_{RPT} models on any of the tasks, the RDN_{RBC}^{Ceil} is significantly higher than RDN_{RPT}^{Ceil} for Cora and IMDb. This indicates that, when there is enough seed information, RDN_{RBC} models may achieve significant performance gains over RDN_{RPT} models.

1.5 Related Work

1.5.1 Probabilistic Relational Models

Probabilistic relational models are one class of models for density estimation in relational datasets. Examples of PRMs include relational Bayesian networks and relational Markov networks.

As outlined in Section 1.3.1, learning and inference in PRMs involve a *data graph* G_D, a *model graph* G_M, and an *inference graph* G_I. All PRMs model data that can be represented as a graph (i.e., G_D). PRMs use different approximation techniques for inference in G_I (e.g., Gibbs sampling, loopy belief propagation [Murphy et al., 1999]), but they all use a similar process for rolling out an inference graph G_I. Consequently, PRMs differ primarily with respect to the representation of the model graph G_M and how that model is learned.

The RBN learning algorithm [Getoor et al., 2001] for the most part uses standard Bayesian network techniques for parameter estimation and structure learning. One notable exception is that the learning algorithm must check for "legal" structures that are guaranteed to be acyclic when rolled out for inference on arbitrary data graphs. In addition, instead of exhaustive search of the space of relational dependencies, the structure learning algorithm uses greedy iterative-deepening, expanding the search in directions where the dependencies improve the likelihood.

The strengths of RBNs include understandable knowledge representations and efficient learning techniques. For relational tasks, with a huge space of possible dependencies, *selective* models are easier to interpret and understand than *non-selective* models. Closed-form parameter estimation techniques allow for efficient structure learning (i.e., feature selection). Also because reasoning with relational models requires more space and computational resources, efficient learning techniques make relational modeling both practical and feasible.

The directed acyclic graph structure is the underlying reason for the efficiency of RBN learning. As discussed in Section 1.1, the acyclicity requirement precludes the learning of arbitrary autocorrelation dependencies and limits the applicability of these models in relational domains. RDN models enjoy the strengths of RBNs (namely, understandable knowledge representation and efficient learning) without being constrained by an acyclicity requirement.

The RMN learning algorithm [Taskar et al., 2002] uses maximum-a-posteriori parameter estimation with Gaussian priors, modifiying Markov network learning techniques. The algorithm assumes that the clique templates are pre-specified and thus does not search for the best structure. Because the user supplies a set of relational dependencies to consider (i.e., clique templates)—it simply optimizes the potential functions for the specified templates.

RMNs are not hampered by an acyclicity constraint, so they can represent and reason with arbitrary forms of autocorrelation. This is particularly important for reasoning in relational datasets where autocorrelation dependencies are nearly ubiquitous and often cannot be structured in an acyclic manner. However, the tradeoff for this increased representational capability is a decrease in learning efficiency. Instead of closed-form parameter estimation, RMNs are trained with conjugate gradient methods, where each iteration requires a round of inference. In large cyclic relational inference graphs, the cost of inference is prohibitively expensive—in particular, without approximations to increase efficiency, feature selection is intractable.

Similar to the comparison with RBNs, RDN models enjoy the strengths of RMNs but not their weaknesses. More specifically, RDNs are able to reason with arbitrary forms of autocorrelation without being limited by efficiency concerns during learning. In fact, the pseudolikelihood estimation technique used by RDNs has been used recently to make feature selection tractable for conditional random field models [McCallum, 2003].

1.5.2 Probabilistic Logic Models

A second class of models for density estimation consists of extensions to conventional logic programming that support probabilistic reasoning in first-order logic environments. We will refer to this class of models as *probabilistic logic models* (PLMs). Examples of PLMs include Bayesian logic programs [Kersting and Raedt, 2002] and Markov logic networks [Richardson and Domingos, 2005].

PLMs represent a joint probability distribution over the groundings of a first-order knowledge base. The first-order knowledge base contains a set of first-order formulae, and the PLM model associates a set of weights/probabilities with each of the formulae. Combined with a set of constants representing objects in the domain, PLM models specify a probability distribution over possible truth assignments to groundings of the first-order formulae. Learning a PLM consists of two tasks: generating the relevant first-order clauses, and estimating the weights/probabilities associated with each clause.

Within this class of models, Markov logic networks (MLN) are most similar in nature to RDNs. In MLNs, each node is a grounding of a predicate in a first-order knowledge base, and features correspond to first-order formulae and their truth values. Learning an MLN consists of estimating the feature weights and selecting which features to include in the final structure. The input knowledge base defines the relevant relational neighborhood, and the algorithm restricts the search by limiting the number of distinct variables in a clause, using a weighted pseudolikelihood scoring function for feature selection [Kok and Domingos, 2005].

MLNs ground out to undirected Markov networks. In this sense, they are quite similar to RMNs, sharing the same strengths and weaknesses—they are capable of representing cyclic autocorrelation relationships but suffer from the complexity of full joint inference during learning, which decreases efficiency. Kok and Domingos [2005] have recently demonstrated the promise of efficient pseudolikelihood structure learning techniques. Our future work will investigate the performance tradeoffs between RDN and MLN approaches to pseudolikelihood estimation for learning.

1.5.3 Collective Inference

Collective inference models exploit autocorrelation dependencies in a network of objects to improve predictions. Joint relational models, such as those discussed above, are able to exploit autocorrelation to improve predictions by estimating joint probability distributions over the entire graph and collectively inferring the labels of related instances.

An alternative approach to collective inference combines local individual classification models (e.g., RBCs) with a joint inference procedure (e.g., relaxation labeling). Examples of this technique include iterative classification [Neville and Jensen, 2000], link-based classification [Lu and Getoor, 2003], and probabilistic relational neighbor [Macskassy and Provost, 2003, 2004]. These approaches to collective inference were developed in an adhoc procedural fashion, motivated by the observation that they appear to work well in practice. RDN models formalize this approach in a principled framework—learning models locally (maximizing psuedolikelihood) and combining them with a global inference procedure (Gibbs sampling) to recover a full joint distribution. In this work we have demonstrated that autocorrelation is the reason behind improved performance in collective inference (see [Jensen et al., 2004] for more detail) and explored the situations under which we can expect this type of approximation to perform well.

1.6 Discussion and Future Work

In this paper we presented relational dependency networks, a new form of probabilistic relational model. We showed the RDN learning algorithm to be a relatively simple method for learning the structure and parameters of a probabilistic graphical model. In addition, RDNs allow us to exploit existing techniques for learning conditional probability distributions of relational datasets. Here we have chosen to exploit our prior work on RPTs, which construct parsimonious models of relational data, and RBCs, which are simple and surprisingly effective non-selective models. We expect the general properties of RDNs to be retained if other approaches to learning conditional probability distributions are used, given that those approaches learn accurate local models.

The primary advantage of RDN models is the ability to efficiently learn and reason with autocorrelation. Autocorrelation is a nearly ubiquitous phenomenon in relational datasets and the dependencies are often cyclic in nature. If a dataset exhibits autocorrelation, and a model can learn the resulting dependencies, then we can exploit those dependencies to improve overall inferences by collectively inferring values for the entire set of instances simultaneously. The real and synthetic data experiments in this paper show that collective inference with RDNs can offer significant improvement over conditional approaches when autocorrelation is present in the data. Except in rare cases, the performance of RDNs approaches the performance that would be possible if all the class labels of related instances were known. Because our analysis indicates that the amount of seed information may interact with the level of autocorrelation and local model characteristics to impact performance, future work will attempt to quantify these effects more formally.

We also presented learned RDNs for a number of real-world relational domains, demonstrating another strength of RDNs—their understandable and intuitive knowledge representation. Comprehensible models are a cornerstone of the knowledge discovery process, which seeks to identify novel and interesting patterns in large datasets. Domain experts are more willing to trust, and make regular use of, understandable models—particularly when the induced models are used to support additional reasoning. Understandable models also aid analysts' assessment of the utility of the additional relational information, potentially reducing the cost of information gathering and storage and the need for data transfer among organizations—increasing the practicality and feasibility of relational modeling.

Future work will compare RDN models to relational Markov networks and Markov logic networks in order to quantify the performance tradeoffs for using pseudolikelihood functions rather than full likelihood functions for both parameter estimation and structure learning, particularly over datasets with varying levels of autocorrelation. Based on theoretical analysis of pseudolikelihood estimation [e.g., Geman and Graffine, 1987], we expect there to be little difference when autocorrelation is low and increased variance when autocorrelation is high. If this is the case, there will need to be enough training data to withstand the increase in variance. Alternatively, bagging techniques may be a means of reducing variance with only a moderate increase in computational cost. In either case, the simplicity and relative efficiency of RDN methods are a clear win for learning models in relational domains.

Acknowledgements

The authors acknowledge the invaluable assistance of A. Shapira, and helpful comments from C. Loiselle.

This effort is supported by DARPA and NSF under contract numbers IIS0326249 and HR0011-04-1-0013. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied of DARPA, NSF, or the U.S. Government.

References

A. Bernstein, S. Clearwater, and F. Provost. The relational vector-space model and industry classification. In *Proceedings of the IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 8–18, 2003.

J. Besag. Statistical analysis of non-lattice data. *The Statistician*, 24:3:179–195, 1975.

H. Blau, N. Immerman, and D. Jensen. A visual query language for relational knowledge discovery. Technical Report 01-28, University of Massachusetts Amherst, Computer Science Department, 2001.

S. Chakrabarti, B. Dom, and P. Indyk. Enhanced hypertext categorization using hyperlinks. In *Proceedings of the ACM SIGMOD International Conference on Management of Data*, pages 307–318, 1998.

C. Cortes, D. Pregibon, and C. Volinsky. Communities of interest. In *Proceedings of the 4th International Symposium of Intelligent Data Analysis*, pages 105–114, 2001.

P. Domingos and M. Richardson. Mining the network value of customers. In *Proceedings of the 7th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 57–66, 2001.

P. Flach and N. Lachiche. 1BC: A first-order Bayesian classifier. In *Proceedings of the 9th International Conference on Inductive Logic Programming*, pages 92–103, 1999.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 1300–1309, 1999.

S. Geman and C. Graffine. Markov random field image models and their applications to computer vision. In *Proceedings of the 1986 International Congress of Mathematicians*, pages 1496–1517, 1987.

L. Getoor, N. Friedman, D. Koller, and A. Pfeffer. Learning probabilistic relational models. In *Relational Data Mining*, pages 307–335. Springer-Verlag, 2001.

D. Heckerman, D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie. Dependency networks for inference, collaborative filtering and data visualization. *Journal of Machine Learning Research*, 1:49–75, 2000.

D. Heckerman, C. Meek, and D. Koller. Probabilistic models for relational data.

Technical Report MSR-TR-2004-30, Microsoft Research, 2004.

M. Jaeger. Relational Bayesian networks. In *Proceedings of the 13th Conference on Uncertainty in Artificial Intelligence*, pages 266–273, 1997.

D. Jensen and J. Neville. Linkage and autocorrelation cause feature selection bias in relational learning. In *Proceedings of the 19th International Conference on Machine Learning*, pages 259–266, 2002.

D. Jensen and J. Neville. Avoiding bias when aggregating relational data with degree disparity. In *Proceedings of the 20th International Conference on Machine Learning*, pages 274–281, 2003.

D. Jensen, J. Neville, and B. Gallagher. Why collective inference improves relational classification. In *Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 593–598, 2004.

K. Kersting. Representational power of probabilistic-logical models: From upgrading to downgrading. In *IJCAI-2003 Workshop on Learning Statistical Models from Relational Data*, pages 61–62, 2003.

K. Kersting and L. De Raedt. Basic principles of learning Bayesian logic programs. Technical Report 174, Institute for Computer Science, University of Freiburg, 2002.

S. Kok and P. Domingos. Learning the structure of Markov logic networks. In *Proceedings of the 22nd International Conference on Machine Learning*, pages 441–448, 2005.

S. Lauritzen and N. Sheehan. Graphical models for genetic analyses. *Statistical Science*, 18:4:489–514, 2003.

Q. Lu and L. Getoor. Link-based classification. In *Proceedings of the 20th International Conference on Machine Learning*, pages 496–503, 2003.

S. Macskassy and F. Provost. A simple relational classifier. In *Proceedings of the 2nd Workshop on Multi-Relational Data Mining, KDD2003*, pages 64–76, 2003.

S. Macskassy and F. Provost. Classification in networked data: A toolkit and a univariate case study. Technical Report CeDER-04-08, Stern School of Business, New York University, 2004.

A. McCallum. Efficiently inducing features of conditional random fields. In *Proceedings of the 19th Conference on Uncertainty in Artificial Intelligence*, pages 403–410, 2003.

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. A machine learning approach to building domain-specific search engines. In *Proceedings of the 16th International Joint Conference on Artificial Intelligence*, pages 662–667, 1999.

K. Murphy, Y. Weiss, and M. Jordan. Loopy belief propagation for approximate inference: An empirical study. In *Proceedings of the 15th Conference on Uncertainty in Artificial Intelligence*, pages 467–479, 1999.

R. Neal. Probabilistic inference using Markov chain Monte Carlo methods. Technical Report CRG-TR-93-1, Dept of Computer Science, University of Toronto, 1993.

J. Neville, O. Şimşek, D. Jensen, J. Komoroske, K. Palmer, and H. Goldberg. Using relational knowledge discovery to prevent securities fraud. In *Proceedings of the 11th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 449–458, 2005.

J. Neville and D. Jensen. Iterative classification in relational data. In *AAAI-2000 Workshop on Learning Statistical Models from Relational Data*, 2000.

J. Neville and D. Jensen. Supporting relational knowledge discovery: Lessons in architecture and algorithm design. In *Proceedings of the Data Mining Lessons Learned Workshop, ICML2002*, pages 57–64, 2002.

J. Neville and D. Jensen. Collective classification with relational dependency networks. In *Proceedings of the 2nd Multi-Relational Data Mining Workshop, KDD2003*, pages 77–91, 2003.

J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning relational probability trees. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 625–630, 2003a.

J. Neville, D. Jensen, and B. Gallagher. Simple estimators for relational Bayesian classifers. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 609–612, 2003b.

C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. In *Proceedings of the 9th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 167–176, 2003.

A. Popescul, L. Ungar, S. Lawrence, and D. Pennock. Statistical relational learning for document mining. In *Proceedings of the 3rd IEEE International Conference on Data Mining*, pages 275–282, 2003.

M. Richardson and P. Domingos. Markov logic networks. *Journal of Machine Learning Research*, to appear, 2005.

S. Sanghai, P. Domingos, and D. Weld. Dynamic probabilistic relational models. In *Proceedings of the 18th International Joint Conference on Artificial Intelligence*, pages 992–1002, 2003.

B. Taskar, P. Abbeel, and D. Koller. Discriminative probabilistic models for relational data. In *Proceedings of the 18th Conference on Uncertainty in Artificial Intelligence*, pages 485–492, 2002.

B. Taskar, E. Segal, and D. Koller. Probabilistic classification and clustering in relational data. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence*, pages 870–878, 2001.

B. Zadrozny and C. Elkan. Obtaining calibrated probability estimates from decision trees and naive Bayesian classifiers. In *Proceedings of the 18th International Conference on Machine Learning*, pages 609–616, 2001.