2007

# Relational Dependency Networks

Jennifer Neville
*Purdue University*

Journal of Machine Learning Research 8 (2007) 653-692 Submitted 9/05; Revised 9/06; Published 3/07

# Relational Dependency Networks

**Jennifer Neville** NEVILLE@CS.PURDUE.EDU
*Departments of Computer Science and Statistics*
*Purdue University*
*West Lafayette, IN 47907-2107, USA*

**David Jensen** JENSEN@CS.UMASS.EDU
*Department of Computer Science*
*University of Massachusetts Amherst*
*Amherst, MA 01003-4610, USA*

**Editor:** Max Chickering

## Abstract

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 paper, we present relational dependency networks (RDNs), graphical models that are 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. In addition, we use synthetic data to explore model performance under various relational data characteristics, showing that RDN learning and inference techniques are accurate over a wide range of conditions.

**Keywords:** Relational learning, probabilistic relational models, knowledge discovery, graphical models, dependency networks, pseudolikelihood estimation.

## 1. Introduction

Many datasets routinely captured by organizations are relational in nature, yet until recently most machine learning research 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). For example, hyperlinked web pages are more likely to share the same topic than randomly selected pages. More formally, we define relational autocorrelation with respect to an attributed graph $G = (V, E)$, where each node $v \in V$ represents an object and each edge $e \in E$ represents a binary relation. Autocorrelation is measured for a set of instance pairs $P_R$ related through paths of length $l$ in a set of edges $E_R$: $P_R = \{(v_i, v_j) : e_{ik_1}, e_{k_1k_2}, ..., e_{k_lj} \in E_R\}$, where $E_R = \{e_{ij}\} \subseteq E$. It is the correlation between the values of a variable $X$ on the instance pairs $(v_i.x, v_j.x)$ such that $(v_i, v_j) \in P_R$. 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 board members (Neville and Jensen, 2000)
- Fraud status of cellular customers who call common numbers (Fawcett and Provost, 1997; Cortes et al., 2001)
- Topics of coreferent scientific papers (Taskar et al., 2001; Neville and Jensen, 2003)
- Functions of colocated proteins 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 news stories (Bernstein et al., 2003)
- Tuberculosis infection among people in close contact (Getoor et al., 2001)
- Product/service adoption among customers in close communication (Domingos and Richardson, 2001; Hill et al., 2006)

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 networks to relational domains, removing the assumption of independent and identically distributed instances that underlies conventional learning techniques.[2] 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[3] (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 dependencies 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 the autocorrelation between two web pages as an undirected correlation. 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.[4]

Undirected PRMs, such as relational Markov networks (RMNs) (Taskar et al., 2002), can represent and reason with arbitrary forms of autocorrelation. However, research on these models has focused primarily on parameter estimation and inference procedures. Current implementations of RMNs do 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.

1. Several previous papers (e.g., Friedman et al., 1999; Getoor et al., 2001) use the term *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. Another class of joint models extend conventional logic programming models to support probabilistic reasoning in first-order logic environments (Kersting and Raedt, 2002; Richardson and Domingos, 2006). We refer to these models as *probabilistic logic models* (PLMs). See Section 5.2 for more detail.
3. 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.
4. The limitation is due to the PRM modeling approach (see Section 3.1), which ties parameters across items of the same type and can produce cycles in the rolled out inference graph. This issue is discussed in more detail in Section 5.1.

In this paper, we outline relational dependency networks (RDNs),[5] an extension of dependency networks (Heckerman et al., 2000) for relational data. RDNs can 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; Richardson and Domingos, 2006). To our knowledge, RDNs are the first PRM capable of *learning* cyclic autocorrelation dependencies. RDNs also 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. RDNs approximate the full joint distribution and thus are not guaranteed to specify a consistent probability distribution. 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 be sufficiently accurate.

We start by reviewing the details of dependency networks for propositional data. Then we describe the general characteristics of PRMs and outline the specifics of RDN learning and inference procedures. We evaluate RDN learning and inference on synthetic datasets, showing that RDN learning is accurate for large to moderate-sized datasets and that RDN inference is comparable, or superior, to RMN inference over a range of data conditions. In addition, we evaluate RDNs on five real-world datasets, presenting learned RDNs for subjective evaluation. Of particular note, all the real-world datasets exhibit multiple autocorrelation dependencies that were automatically discovered by the RDN learning algorithm. We evaluate the learned models in a prediction context, where only a single attribute is unobserved, and show that the models outperform conventional conditional models on all five tasks. Finally, we review related work and conclude with a discussion of future directions.

## 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, DNs are not guaranteed to specify a *consistent*[6] joint distribution, where each CPD can be derived from the joint distribution using the rules of probability. This limits the applicability of exact inference techniques. In addition, the correlational DN representation precludes DNs from being used to infer causal relationships. 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

5. This paper continues our previous work on RDNs (Neville and Jensen, 2004).
6. In this paper, we use the term *consistent* to refer to the consistency of the individual CPDs (as Heckerman et al., 2000), rather than the asymptotic properties of a statistical estimator.

CPDs. We begin by reviewing traditional graphical models and then outline the details of dependency networks in this context.

Consider the set of variables $\mathbf{X} = (X_1, ..., X_n)$ over which we would like to model the joint distribution $p(\mathbf{x}) = p(x_1, ..., x_n)$. We use upper case letters to refer to random variables and lower case letters to refer to an assignment of values to the variables.

A Bayesian network for $\mathbf{X}$ uses a directed acyclic graph $G = (V, E)$ and a set of conditional probability distributions $P$ to represent the joint distribution over $\mathbf{X}$. Each node $v \in V$ corresponds to an $X_i \in \mathbf{X}$. The edges of the graph encode dependencies among the variables and can be used to infer conditional independence among variables using notions of d-separation. The parents of node $X_i$, denoted $PA_i$, are the set of $v_j \in V$ such that $(v_j, v_i) \in E$. The set $P$ contains a conditional probability distribution for each variable given its parents, $p(x_i|pa_i)$. The acyclicity constraint on $G$ ensures that the CPDs in $P$ factor the joint distribution into the formula below. A directed graph is acyclic if there is no directed path that starts and ends at the same variable. More specifically, there can be no self-loops from a variable to itself. Given $(G, P)$, the joint probability for a set of values $\mathbf{x}$ is computed with the formula:

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|pa_i)$$

A Markov network for $\mathbf{X}$ uses an undirected graph $U = (V, E)$ and a set of potential functions $\Phi$ to represent the joint distribution over $\mathbf{X}$. Again, each node $v \in V$ corresponds to an $X_i \in \mathbf{X}$ and the edges of the graph encode conditional independence assumptions. However, with undirected graphs, conditional independence can be inferred using simple graph separation. Let $C(U)$ be the set of cliques in the graph $U$. Then each clique $c \in C(U)$ is associated with a set of variables $X_c$ and a clique potential $\phi_c(x_c)$ which is a non-negative function over the possible values for $x_c$. Given $(U, \Phi)$, the joint probability for a set of values $\mathbf{x}$ is computed with the formula:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{i=1}^{c} \phi_i(x_{c_i})$$

where $Z = \sum_X \prod_{i=1}^{c} \phi_i(x_{c_i})$ is a *normalizing constant*, which sums over all possible instantiations of $\mathbf{x}$ to ensure that $p(\mathbf{x})$ is a true probability distribution.

### 2.1 DN Representation

Dependency networks are an alternative form of graphical model that approximates 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 directed 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 $v_i$ ($(v_j, v_i) \in E$ 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 Section 3.4 for details). From the joint distribution, we can extract any probabilities of interest.

For example, the DN in Figure 1 models the set of variables: $\mathbf{X} = \{X_1, X_2, X_3, X_4, X_5\}$. Each node is conditionally independent of the other nodes in the graph given its immediate neighbors (e.g., $X_1$ is conditionally independent of $\{X_2, X_4\}$ given $\{X_3, X_5\}$). Each node contains a CPD, which specifies a probability distribution over its possible values, given the values of its parents.

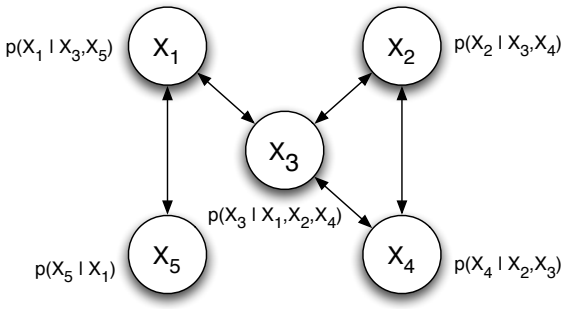


Figure 1: Example dependency network.

### 2.2 DN Learning

Both the structure and parameters of DNs 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 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 all available variables.

### 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 the parameter estimates. If this is the case, the joint distribution derived numerically from the CPDs will not sum to one. However, 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). 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.

## 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 it 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.[7] 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.

### 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 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 2a. 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).[8] Each node $v_i \in V_D$ and edge $e_j \in E_D$ is associated with a type, $T(v_i) = t_{v_i}$ and $T(e_j) = t_{e_j}$ (e.g., paper, cited-by). Each item[9] type $t \in T$ has a number of associated attributes $\mathbf{X^t} = (X_1^t, ..., X_m^t)$ (e.g., topic, year). Consequently, each object $v_i$ and link $e_j$ is associated with a set of attribute values determined by their type,

7. See Section 5.1 for a more detailed description of RBNs and RMNs.
8. We use rectangles to represent objects, circles to represent random variables, dashed lines to represent relations, and solid lines to represent probabilistic dependencies.
9. We use the generic term "item" to refer to objects or links.

$\mathbf{X}_{\mathbf{v_i}}^{\mathbf{t_{v_i}}} = (X_{v_i1}^{t_{v_i}}, ..., X_{v_im}^{t_{v_i}})$ and $\mathbf{X}_{\mathbf{e_j}}^{\mathbf{t_{e_j}}} = (X_{e_j1}^{t_{e_j}}, ..., X_{e_jm'}^{t_{e_j}})$. A PRM represents a joint distribution over the values of the attributes in the data graph, $\mathbf{x} = \{\mathbf{x}_{\mathbf{v_i}}^{\mathbf{t_{v_i}}} : v_i \in V \; s.t. \; T(v_i) = t_{v_i}\} \cup \{\mathbf{x}_{\mathbf{e_j}}^{\mathbf{t_{e_j}}} : e_j \in E \; s.t. \; T(e_j) = t_{e_j}\}$.

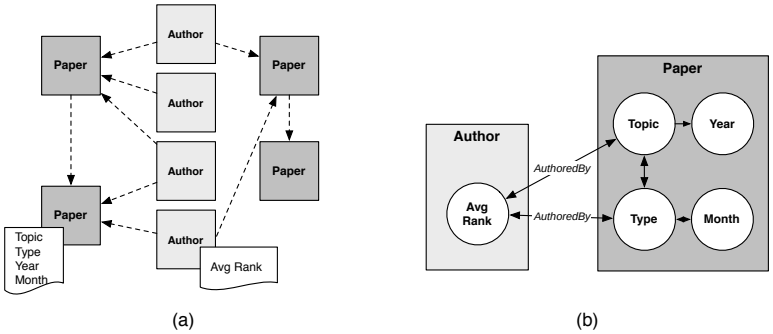


Figure 2: Example (a) data graph and (b) model graph.

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. The relations in $G_D$ are used to limit the search for possible statistical dependencies, thus they constrain the set of edges that can appear in $G_M$. However, note that a relationship between two objects in $G_D$ does not necessarily imply a probabilistic dependence between their attributes in $G_M$.

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 items of 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 PRMs use aggregation functions to generalize across heterogeneous attributes sets (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 2b.[10] It models the data in Figure 2a, 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 represented by arcs within a rectangle; arcs that cross rectangle boundaries represent dependencies among attributes of related objects, with edge labels indicating the underlying relations. For example, $month_i$ depends on $type_i$, while $avgrank_j$ depends on the $type_k$ and $topic_k$ for all papers $k$ written by author $j$ in $G_D$.

There is a nearly limitless range of dependencies that could be considered by algorithms for learning PRMs. 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 an 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 in $G_D$ (e.g., attributes of items up to $k$ links away), or (2) greedy iterative-deepening search, expanding the search in $G_D$ 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 "roll out." The roll out 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 $G_D$ determine 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 3 shows the model from Figure 2b rolled out over the dataset in Figure 2a. 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.

### 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 a directed 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 $p(x_k^t \mid pa_{x_k^t})$. Figure 2b illustrates an example RDN model graph for the data graph in Figure 2a. The graphical representation illustrates the qualitative component ($G_D$) of the RDN—it does not depict

10. For clarity, we omit cyclic autocorrelation dependencies in this example. See Section 4.2 for more complex model graphs.

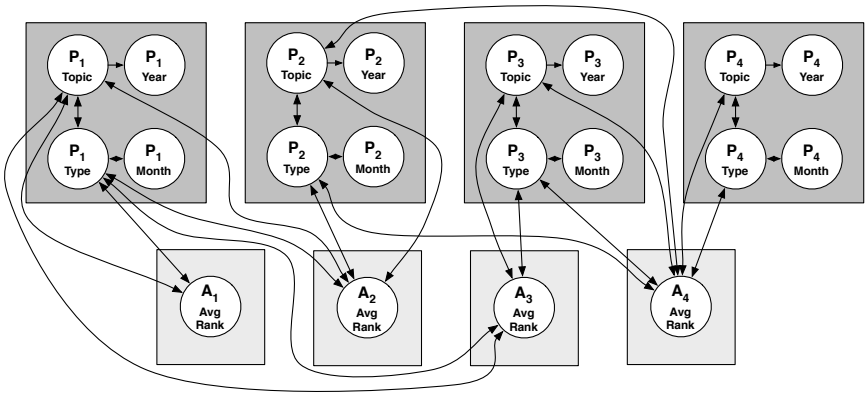


Figure 3: Example inference graph.

the quantitative component ($P$) of the model, which consists of CPDs that use aggregation functions. Although conditional independence is inferred using an undirected view of the graph, directed edges are useful for representing the set of variables in each CPD. For example, in Figure 2b the CPD for *year* contains *topic* but the CPD for *topic* does not contain *year*. This represents any inconsistencies that result from the RDN learning technique.

A *consistent* RDN specifies a joint probability distribution $p(\mathbf{x})$ over the attribute values of a relational dataset from which each CPD $\in P$ can be derived using the rules of probability. There is a direct correspondence between consistent RDNs and relational Markov networks. It is similar to the correspondence between consistent DNs and Markov networks (Heckerman et al., 2000), but the correspondence is defined with respect to the template model graphs $G_M$ and $U_M$.

**Theorem 1** *The set of positive distributions that can be encoded by a consistent RDN* $(G_M, P)$ *is equal to the set of positive distributions that can be encoded by an RMN* $(U_M, \Phi)$ *provided (1)* $G_M = U_M$*, and (2)* $P$ *and* $\Phi$ *use the same aggregation functions.*

**Proof** Let $p$ be a positive distribution defined by an RMN $(U_M, \Phi)$ for $G_D$. First, we construct a Markov network with tied clique potentials by rolling out the RMN inference graph $U_I$ over the data graph $G_D$. By Theorem 1 of Heckerman et al. (2000), which uses the Hammersley-Clifford theorem (Besag, 1974), there is a corresponding dependency network that represents the same distribution $p$ as the Markov network $U_I$. Since the conditional probability distribution for each occurrence of an attribute $k$ of a given type $t$ (i.e., $\forall i\ (v_i \in V_D\ \vee\ e_i \in E_D)\ \wedge\ T(i) = t\ \ p(x^t_{ik}|\mathbf{x})$) is derived from the Markov network, we know that the resulting CPDs will be identical—the nodes adjacent to each occurrence are equivalent by definition, thus by the global Markov property the derived CPDs will be identical. From this dependency network we can construct a consistent RDN $(G_M, P)$ by first setting $G_M = U_M$. Next, we compute from $U_I$ the CPDs for the attributes of each item type: $p(x^t_k|\mathbf{x} - \{x^t_k\})$ for $t \in T, X^t_k \in \mathbf{X^t}$. To derive the CPDs for $P$, the CPDs must use the same aggregation functions as the potentials in $\Phi$. Since the adjacencies in the RDN model graph are the same as those in the RMN model graph, and there is a correspondence between the rolled out DN and MN, the distribution encoded by the RDN is $p$.

Next let $p$ be a positive distribution defined by an RDN $(G_M, P)$ for $G_D$. First, we construct a dependency network with tied CPDs by rolling out the RDN inference graph $G_I$ over the data graph $G_D$. Again, by Theorem 1 of Heckerman et al. (2000), there is a corresponding Markov network that represents the same distribution $p$ as the dependency network $G_I$. Of the valid Markov networks representing $p$, there will exist a network where the potentials are tied across occurrences of the same clique template (i.e., $\forall c_i \in C \;\; \phi_C(x_C)$). This follows from the first part of the proof, which shows that each RMN with tied clique potentials can be transformed to an RDN with tied CPDs. From this Markov network we can construct an RMN $(U_M, \Phi)$ by setting $U_M = G_M$ and grouping the set of clique template potentials in $\Phi$. Since the adjacencies in the RMN model graph are the same as those in the RDN model graph, and since there is a correspondence between the rolled out MN and DN, the distribution encoded by the RMN is $p$. ■

This proof shows an exact correspondence between *consistent* RDNs and RMNs. We cannot show the same correspondence for general RDNs. However, we will show in Section 3.4 that Gibbs sampling can be used to extract a unique joint distribution, regardless of the consistency of the model.

### 3.3 RDN Learning

Learning a PRM 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 RBNs. 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, it is possible for RMN techniques to learn cyclic autocorrelation dependencies in principle. However, 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 *pseudolikelihood* techniques (Besag, 1975), which were developed for modeling spatial datasets with similar autocorrelation dependencies. The pseudolikelihood for data graph $G_D$ is computed as a product over the item types $t$, the attributes of that type $X^t$, and the items of

that type $v, e$:

$$PL(G_D;\theta) = \prod_{t \in T} \prod_{X_i^t \in \mathbf{X}^t} \prod_{v:T(v)=t} p(x_{vi}^t | pa_{x_{vi}^t};\theta) \prod_{e:T(e)=t} p(x_{ei}^t | pa_{x_{ei}^t};\theta) \quad (1)$$

On the surface, Equation 1 may appear similar to a likelihood that specifies a joint distribution of an RBN. However, the CPDs in the RDN 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 CPDs for variables in $PA_{X_{vi}^t}$ was conditioned on $X_{vi}^t$. The parents of $X_{vi}^t$ may include other variables on the same item (e.g., $X_{vi'}^t$ such that $i' \neq i$), the same variable on related items (e.g., $X_{v'i}^t$ such that $v' \neq v$), or other variables on related items (e.g., $X_{v'i'}^{t'}$ such that $v' \neq v$ and $i' \neq i$).

Pseudolikelihood estimation avoids the complexities of estimating $Z$ and the requirement of acyclicity. Instead of optimizing the log-likelihood of the full joint distribution, we optimize the pseudo-loglikelihood. The contribution for each variable is conditioned on all other attribute values in the data, thus we can maximize the pseudo-loglikelihood for each variable independently:

$$log\ PL(G_D;\theta) = \sum_{t \in T} \sum_{X_i^t \in X^t} \sum_{v:T(v)=t} log\ p(x_{vi}^t | pa_{x_{vi}^t};\theta) + \sum_{e:T(e)=t} log\ p(x_{ei}^t | pa_{x_{ei}^t};\theta)$$

In addition, this approach can make use of 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).

Maximizing the pseudolikelihood function gives the maximum pseudolikelihood estimate (MPLE) of $\theta$. To estimate the parameters we need to solve the following pseudolikelihood equation:

$$\frac{\partial}{\partial \theta} PL(G_D;\theta) = 0 \quad (2)$$

With this approach we lose the asymptotic efficiency properties of maximum likelihood estimators. However, under some general conditions the asymptotic properties of the MPLE can be established. In particular, in the limit as sample size grows, the MPLE will be an unbiased estimate of the true parameter $\theta_0$ and it will be normally distributed. Geman and Graffine (1987) established the first proof of the properties of maximum pseudolikelihood estimators of fully observed data. Gidas (1986) gives an alternative proof and Comets (1992) establishes a more general proof that does not require a finite state space $\mathbf{x}$ or stationarity of the true distribution $P_{\theta_0}$.

**Theorem 2** *Assume the following regularity conditions*[11] *are satisfied for an RDN:*

1. *The model is identifiable (i.e., if $\theta \neq \theta'$, then $PL(G_D;\theta) \neq PL(G_D;\theta')$).*

11. These are the standard regularity conditions (e.g., Casella and Berger, 2002) used to prove asymptotic properties of estimators, which are satisfied in most reasonable problems.

2. *The distributions $PL(G_D;\theta)$ have common support and are differentiable with respect to $\theta$.*

3. *The parameter space $\Omega$ contains an open set $\omega$ of which the true parameter $\theta_0$ is an interior point.*

*In addition, assume the pseudolikelihood equation (Equation 2) has a unique solution in $\Omega$ almost surely as $|G_D| \to \infty$. Then, provided that $G_D$ is of bounded degree, the MPLE $\tilde{\theta}$ converges in probability to the true value $\theta_0$ as $|G_D| \to \infty$.*

**Proof** Provided the size of the RDN does not grow as the size of the dataset grows (i.e., $|P|$ remains constant as $|G_D| \to \infty$) and $G_D$ is of bounded degree, then previous proofs apply. We provide the intuition for the proof here and refer the reader to Comets (1992); White (1994); Lehmann and Casella (1998) for details. Let $\tilde{\theta}$ be the maximum pseudolikelihood estimate that maximizes $PL(G_D;\theta)$. As $|G_D| \to \infty$, the data will consist of all possible data configurations for each CPD $\in P$ (assuming bounded degree structure in $G_D$). As such, the pseudolikelihood function will converge to its expectation, $PL(G_D;\theta) \to E(PL(G_D;\theta))$. The expectation is maximized by the true parameter $\theta_0$ because the expectation is taken with respect to all possible data configurations. Therefore as $|G_D| \to \infty$, the MPLE converges to the true parameter (i.e., $\tilde{\theta} - \theta_0 \to 0$). ■

The RDN learning algorithm is similar to the DN learning algorithm, except we use a relational probability estimation algorithm to learn the set of conditional models, maximizing pseudolikelihood for each variable separately. The algorithm input consists of: (1) $G_D$: a relational data graph, (2) $R$: a conditional relational learner, and (3) $\mathbf{Q^t}$: a set of queries[12] that specify the relational neighborhood considered in $R$ for each type $T$.

Table 1 outlines the learning algorithm in pseudocode. The algorithm cycles over each attribute of each item type and learns a separate CPD, conditioned on the other values in the training data. We discuss details of the subcomponents (querying and relational learners) in the sections below.

The asymptotic complexity of RDN learning is $O(|\mathbf{X}| \cdot |PA_X| \cdot N)$, where $|\mathbf{X}|$ is the number of CPDs to be estimated, $|PA_X|$ is the number of attributes and $N$ is the number of instances, used to estimate the CPD for $X$.[13] Quantifying the asymptotic complexity of RBN and RMN learning is difficult due to the use of heuristic search and numerical optimization techniques. RBN learning requires multiple rounds of parameter estimation during the algorithm's heuristic search through the model space, and each round of parameter estimation has the same complexity as RDN learning, thus RBN learning will generally require more time. For RMN learning, there is no closed-form parameter estimation technique. Instead the models are trained using conjugate gradient, where each iteration requires approximate inference over the unrolled Markov network. In general this RMN nested loop of optimization and approximation will require more time to learn than an RBN (Taskar et al.,

12. Our implementation employs a set of user-specified queries to limit the search space considered during learning. However, a simple depth limit (e.g., $\leq 2$ links away in the data graph) can be used to limit the search space as well.

13. This assumes the complexity of the relational learner $R$ is $O(|PA_X| \cdot N)$, which is true for the two relational learners considered in this paper.

2002). Therefore, given equivalent search spaces, RMN learning is generally more complex than RBN learning, and RBN learning is generally more complex than RDN learning.

---

**Learn RDN** $(G_D, R, \mathbf{Q^t})$:
$P \leftarrow \emptyset$
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 in the relational neighborhood defined by $Q^t$.
    $P \leftarrow P \cup CPD_{X_k^t}$
Use $P$ to form $G_M$.

---

Table 1: RDN learning algorithm.

#### 3.3.1 Queries

In our implementation, we use queries to specify the relational neighborhoods that will be considered by the conditional learner $R$. The queries' structures define 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 4a.[14] 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. Note the constraint on paper ID in the lower left corner—this ensures that the target paper does not match as a reference in the resulting subgraphs. Figure 4b 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 learner $R$ 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 simple 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).

14. We have modified QGraph's visual representation to conform to our convention of using rectangles to represent objects and dashed lines to represent relations.

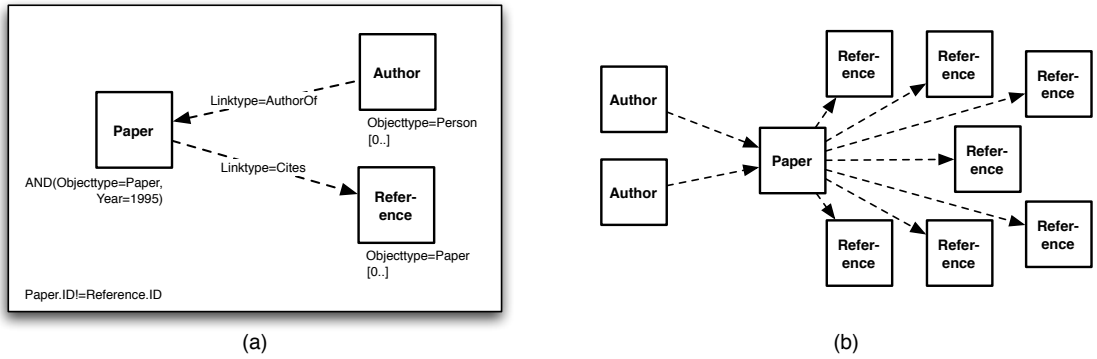


Figure 4: (a) Example QGraph query: Textual annotations specify match conditions on attribute values; numerical annotations (e.g., [0..]) specify constraints on the cardinality of matched objects (e.g., zero or more authors), and (b) matching subgraph.

### 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 $G_M$ appropriately. If $R$ selects all of the available attributes, the RDN will be fully connected.

In principle, any conditional relational learner can be used as a subcomponent to learn the individual CPDs provided that it can closely approximate CPDs consistent with the joint distribution. 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. RBCs 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.[15] 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 $t \in T$, the query scope specifies the set of item types $\mathbf{T_R}$ that form the relevant relational neighborhood for $t$. Note that $\mathbf{T_R}$ does not necessarily contain all item types in the database and the query may also dynamically introduce new types in the returned view of the database (e.g., papers → papers *and* references). For example, in Figure 4a, $t = paper$ and $\mathbf{T_R} = \{paper, author, reference, authorof, cites\}$. To estimate

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

the CPD for attribute $X$ on items $t$ (e.g., paper topic), the RBC considers all the attributes associated with the types in $\mathbf{T_R}$. RBCs are non-selective models, thus all 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. RPTs 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 5 provides an example RPT learned on citation data.

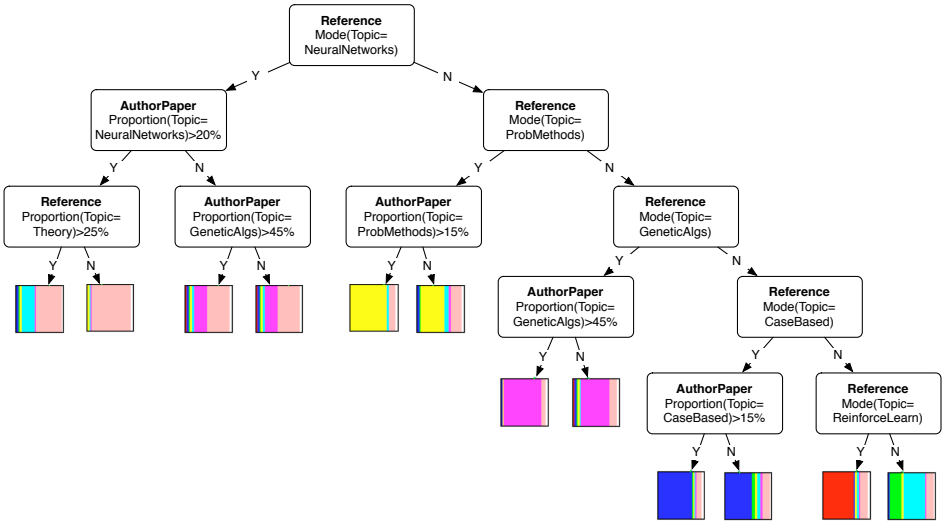


Figure 5: Example RPT to predict machine-learning paper topic.

The RPT algorithm automatically constructs and searches over aggregated relational features to model the distribution of the target variable $X$ on items of type $t$. The algorithm constructs features from the attributes associated with the types $\mathbf{T_R}$ specified in the query for $t$. The algorithm considers four classes of aggregation functions to group multiset values: *mode*, *count*, *proportion*, and *degree* (i.e., the number of values in the multiset). 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., $proportion(A) > 10\%$). All experiments reported herein considered 10 thresholds and discretizations per feature.

The RPT algorithm uses recursive greedy partitioning, splitting on the feature that maximizes the correlation between the feature and the class. Feature scores are calculated using the chi-square statistic and the algorithm uses pre-pruning in the form of a $p$-value cutoff and a depth cutoff to limit tree size and overfitting. All experiments reported herein used *p-value cutoff*=0.05/|*attributes*|, *depth cutoff*=7. Although the objective function does not optimize pseudolikelihood directly, probability estimation trees can be used effectively to approximate CPDs consistent with the underlying joint distribution (Heckerman et al., 2000).

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 RDNs and have a direct impact on inference efficiency because smaller trees limit the size of the final inference graph.

### 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 $\sum_{v \in V_D} |\mathbf{X}^{\mathbf{T(v)}}| + \sum_{e \in E_D} |\mathbf{X}^{\mathbf{T(e)}}|$. Roll out 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 roll out copies of model templates (e.g., hidden Markov models, conditional random fields (Lafferty et al., 2001)).

We use Gibbs samplers for inference in RDNs. This refers to a procedure where a random ordering of the variables is selected; each variable is initialized to an arbitrary value; and then each variable is visited (repeatedly) in order, where its value is resampled according to its conditional distribution. Gibbs sampling can be used to extract a unique joint distribution, regardless of the consistency of the model.

**Theorem 3** *The procedure of a Gibbs sampler applied to an RDN* $(G, P)$*, where each* $X_i$ *is discrete and each local distribution in* $P$ *is positive, defines a Markov chain with a unique stationary joint distribution* $\tilde{\pi}$ *for* $\mathbf{X}$ *that can be reached from any initial state of the chain.*

**Proof** The proof that Gibbs sampling can be used to estimate the joint distribution of a dependency network (Heckerman et al., 2000) applies to rolled out RDNs as well. We restate the proof here for completeness.

Let $\mathbf{x}^t$ be the sample of $\mathbf{x}$ after the $t^{th}$ iteration of the Gibbs sampler. The sequence $\mathbf{x}^1, \mathbf{x}^2, ...$ can be viewed as samples drawn from a homogeneous Markov chain with transition matrix $\tilde{\mathbf{P}}$, where $\tilde{\mathbf{P}}_{ij} = p(\mathbf{x}^{t+1} = j | \mathbf{x}^t = i)$. The matrix $\tilde{\mathbf{P}}$ is the product $\tilde{\mathbf{P}}^1 \cdot \tilde{\mathbf{P}}^2 \cdot ... \cdot \tilde{\mathbf{P}}^n$, where $\tilde{\mathbf{P}}^k$ is the *local* transition matrix describing the resampling of $X^k$ according to the

local distribution of $p(x_k|\mathbf{pa}_k)$. The positivity of the local distributions guarantees the positivity of $\tilde{\mathbf{P}}$. The positivity of $\tilde{\mathbf{P}}$ in turn guarantees that the Markov chain is irreducible and aperiodic. Consequently there exists a unique joint distribution that is stationary with respect to $\tilde{\mathbf{P}}$, and this stationary distribution can be reached from any starting point. ■

This shows that a Gibbs sampling procedure can be used with an RDN to recover samples from a unique stationary distribution $\tilde{\pi}$, but how close will this distribution be to the true distribution $\pi$? Small perturbations in the local CPDs could propagate in the Gibbs sampling procedure to produce large deviations in the stationary distribution. Heckerman et al. (2000) provide some initial theoretical analysis that suggests that Markov chains with good convergence properties will be insensitive to deviations in the transition matrix. This implies that when Gibbs sampling is effective (i.e., converges), then $\tilde{\pi}$ will be close to $\pi$ and the RDN will be a close approximation to the full joint distribution.

Table 2 outlines the inference algorithm. To estimate a joint distribution, we start by rolling out the model $G_M$ onto the target dataset $G_D$ and forming the inference graph $G_I$. The values of all unobserved variables are initialized to values drawn from prior distributions, which we estimate empirically from the training set. 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 assignment of values of $X$ (e.g., when $|X|$ is large), it is often reasonably fast to use Gibbs sampling 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. Section 4.1 includes convergence graphs for synthetic data experiments.

## 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 with reasonable dataset sizes and that the model is robust to varying levels of autocorrelation. In addition, to assess the quality of the RDN approximation for inference, we compare RDNs to RMNs, showing that RDNs achieve equivalent or better performance over a range of datasets. Next, we learn RDNs of five real-world datasets to illustrate the types of domain knowledge that the models discover automatically. In addition, we evaluate RDNs 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.

**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.* $T(v_i) = t$ and $e_i \in E_D$ *s.t.* $T(e_i) = t$:
      $V_I \leftarrow V_I \cup \{X_{ik}^t\}$
    For each $v_i \in V_D$ *s.t.* $T(v_i) = t$ and $e_i \in E_D$ *s.t.* $T(e_i) = t$:
      For each $v_j \in V_D$ *s.t.* $X_{v_j} \in pa_{X_{ik}^t}$ and each $e_j \in E_D$ *s.t.* $X_{e_j} \in pa_{X_{ik}^t}$:
        $E_I \leftarrow E_I \cup \{e_{ij}\}$

For each $v \in V_I$: \\ *initialize Gibbs sampling*
  Randomly initialize $x_v$ to value drawn from prior distribution $p(x_v)$

$S \leftarrow \emptyset$ \\ *Gibbs sampling procedure*
Choose a random ordering over $V_I$
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

Table 2: RDN inference algorithm.

### 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 an autocorrelated class label and linkage due to an underlying (hidden) group structure. Each object has four boolean attributes: $X_1$, $X_2$, $X_3$, and $X_4$. We used the following generative process for a dataset with $N_O$ objects and $N_G$ groups:

For each object $i$, $1 \leq i \leq N_O$:
  Choose a group $g_i$ uniformly from the range $[1, N_G]$.
For each object $j$, $1 \leq j \leq N_O$:
  For each object $k$, $j < k \leq N_O$:
    Choose whether the two objects are linked from $p(E|G_j = G_k)$, a Bernoulli probability conditioned on whether the two objects are in the same group.
For each object $i$, $1 \leq i \leq N_O$:
  Randomly initialize the values of $\mathbf{X} = \{X_1, X_2, X_3, X_4\}$ from a uniform prior distribution.
Update the values of $\mathbf{X}$ with 500 iterations of Gibbs sampling using $RDN^*$, a manually specified model.[16]

The data generation procedure for $\mathbf{X}$ uses a manually specified model 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 conditional models for $p(X_1|\mathbf{X}_{\mathbf{1}R}, X_2, X_3, X_4)$. $RPT_{0.5}$ refers to the RPT CPD that is used to generate data with autocorrelation levels of 0.5. $RBC_{0.5}$ refers to the analogous RBC CPD. Appendix A contains detailed specifications of these models. Unless otherwise specified, the experiments use the settings below:

$$
\begin{aligned}
N_O &= 250 \\
N_G &= \frac{N_O}{10} \\
p(E|G_j{=}G_k) &= \{p(E{=}1|G_j{=}G_k) = 0.50;\ p(E{=}1|G_j{\neq}G_k) = \frac{1}{N_O}\} \\
RDN^* &=: [\, p(X_1|\mathbf{X}_{\mathbf{1}R}, X_2, X_3, X_4) = p(X_1|\mathbf{X}_{\mathbf{1}R}, X_2) = RPT_{0.5} \textit{ or } RBC_{0.5}; \\
&\quad\ p(X_2|X_1) = \{p(X_2{=}1|X_1{=}1) = p(X_2{=}0|X_1{=}0) = 0.75\}; \\
&\quad\ p(X_3 = 1) = p(X_4 = 1) = 0.50\,]
\end{aligned}
$$

#### 4.1.1 RDN Learning

The first set of synthetic experiments examines the effectiveness of the RDN learning algorithm. We learned CPDs for $X_1$ using the intrinsic attributes of the object $(X_2, X_3, X_4)$ as well as the class label of directly related objects $(X_{1R})$. We also learned CPDs for each

16. We will use a star (i.e., $RDN^*$) to denote manually-specified RDNs.

attribute $(X_2, X_3, X_4)$ using the class label $(X_1)$. This mimics the structure of the true model used for data generation (i.e., $RDN^*$).

We compared two different learned RDNs: $RDN_{RBC}$ uses RBCs for the component learner $R$; $RDN_{RPT}$ uses RPTs 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 parameter 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.

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 models ($RDN^*_{RPT}$, $RDN^*_{RBC}$) and the learned models ($RDN_{RPT}$, $RDN_{RBC}$). 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 6 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 $p(X_1|\mathbf{X}_{\mathbf{1}R}, X_2)$ 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. We plot the mean pseudolikelihood for both the learned models and the true models. The top row reports experiments with data generated from an $RDN^*_{RPT}$, where we learned an $RDN_{RPT}$. The bottom row reports experiments with data generated from an $RDN^*_{RBC}$, where we learned an $RDN_{RBC}$.

These experiments show that the learned $RDN_{RPT}$ is 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.

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. One possible source of error is variance due to lack of selectivity in the $RDN_{RBC}$, 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).

#### 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

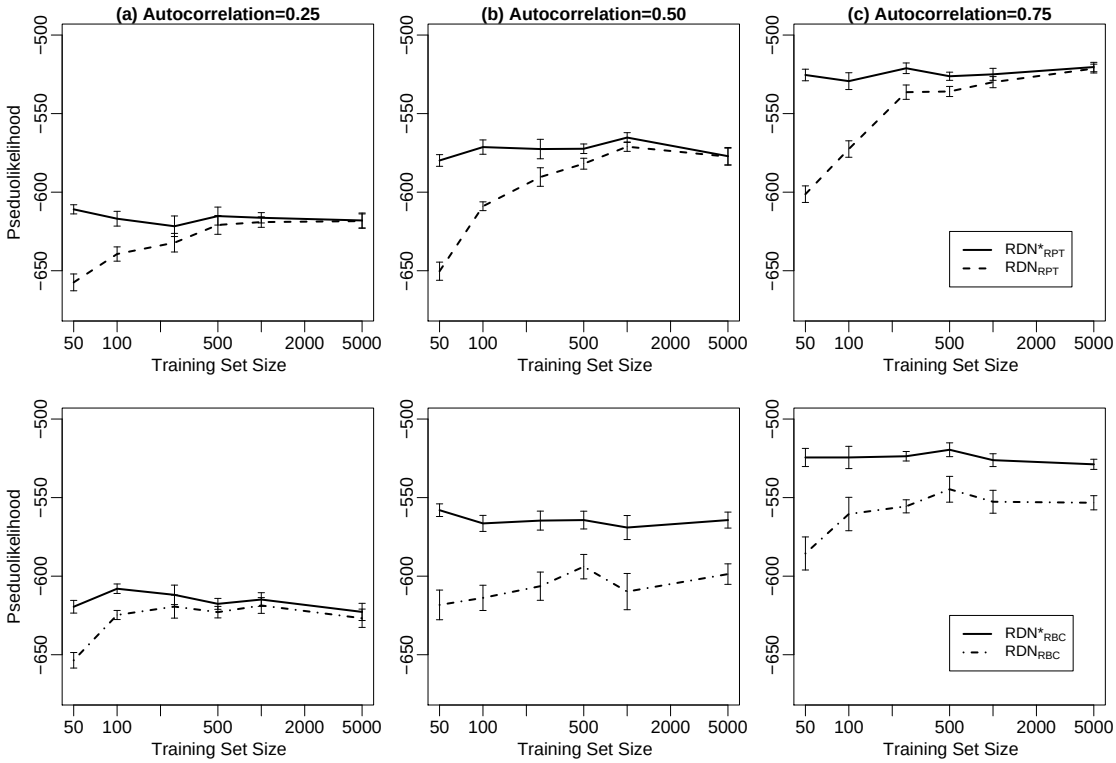


Figure 6: Evaluation of RDN learning.

with the $RDN^*_{RPT}$ and $RDN^*_{RBC}$ as described above and learned models for $X_1$ using the intrinsic attributes of the object $(X_2, X_3, X_4)$ as well as the class label and the attributes of directly related objects $(X_{1R}, X_{2R}, X_{3R}, X_{4R})$. At each autocorrelation level, we generated 10 training sets (size 500) to learn the models. For each training set, we generated 10 test sets (size 250) and used the learned models to infer marginal probabilities for $X_1$ on the test set instances. To evaluate the predictions, we report area under the ROC curve (AUC).[17] These experiments used the same levels of autocorrelation outlined above.

We compare the performance of four types of models. First, we measure the performance of RPTs and RBCs. These are *conditional models* that reason about each instance independently and do not use the class labels of related instances. Second, we measure the performance of learned RDNs: $RDN_{RBC}$ and $RDN_{RPT}$. For RDN inference, we used fixed-length Gibbs chains of 2000 samples with burn-in of 100. Third, we measure performance of the learned RDNs 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*: $RDN^{ceil}_{RBC}$ and $RDN^{ceil}_{RPT}$. Fourth, we measure the performance of two RMNs described below.

The first RMN is non-selective. We construct features from all the attributes available to the RDNs, defining clique templates for each pairwise combination of class label value and

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

attribute value. More specifically, the available attributes consist of the intrinsic attributes of objects, and both the class label and attributes of directly related objects. The second RMN, which we refer to as $RMN_{Sel}$, is a hybrid selective model—clique templates are only specified for the set of attributes selected by the RDN during learning. For both models, we used maximum-a-posteriori parameter estimation to estimate the feature weights, using conjugate gradient with zero-mean Gaussian priors, and a uniform prior variance of 5.[18] For RMN inference, we used loopy belief propagation (Murphy et al., 1999).

We do not compare directly to RBNs because their acyclicity constraint prevents them from representing the autocorrelation dependencies in this domain. Instead, we include the performance of conditional models, which also cannot represent the autocorrelation of $X_1$. Although RBNs and conditional models cannot represent the autocorrelation directly, they can exploit the autocorrelation indirectly by using the observed 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 pages are autocorrelated, then the models can exploit autocorrelation dependencies by modeling the contents of a webpage's neighboring pages. Recent work has shown that collective models (e.g., RDNs) are a low-variance means of reducing bias through direct modeling of the autocorrelation dependencies (Jensen et al., 2004). Models that exploit autocorrelation dependencies indirectly by modeling the observed attributes of related instances, experience a dramatic increase in variance as the number of observed 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 7 graphs AUC results for each model as the proportion of known class labels is varied.

The data for the first set of experiments (top row) were generated with an $RDN^*_{RPT}$. 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 an $RDN_{RPT}$. $RDN_{RPT}$ performance is indistinguishable from that of $RDN^{ceil}_{RPT}$ except when autocorrelation is high and there are no labels to seed inference. In this situation, the predictive attribute values (i.e., $X_2$) are the only information available to constrain the system during inference so the model cannot fully exploit the autocorrelation dependencies. When there is no information to anchor the predictions, there is an identifiability problem—symmetric labelings that are highly autocorrelated, but with opposite values, appear equally likely. In situations where there is little seed information (either attributes or class labels), identifiability problems can increase variance and bias $RDN$ performance towards random.

When there is low or moderate autocorrelation, $RDN_{RPT}$ performance is significantly higher than both RMNs. In these situations, poor RMN performance is likely due to a mismatch in feature space with the data generation model—if the RMN features cannot represent the data dependencies that are generated with aggregated features, the inferred

18. We experimented with a range of priors; this parameter setting produced the best empirical results.

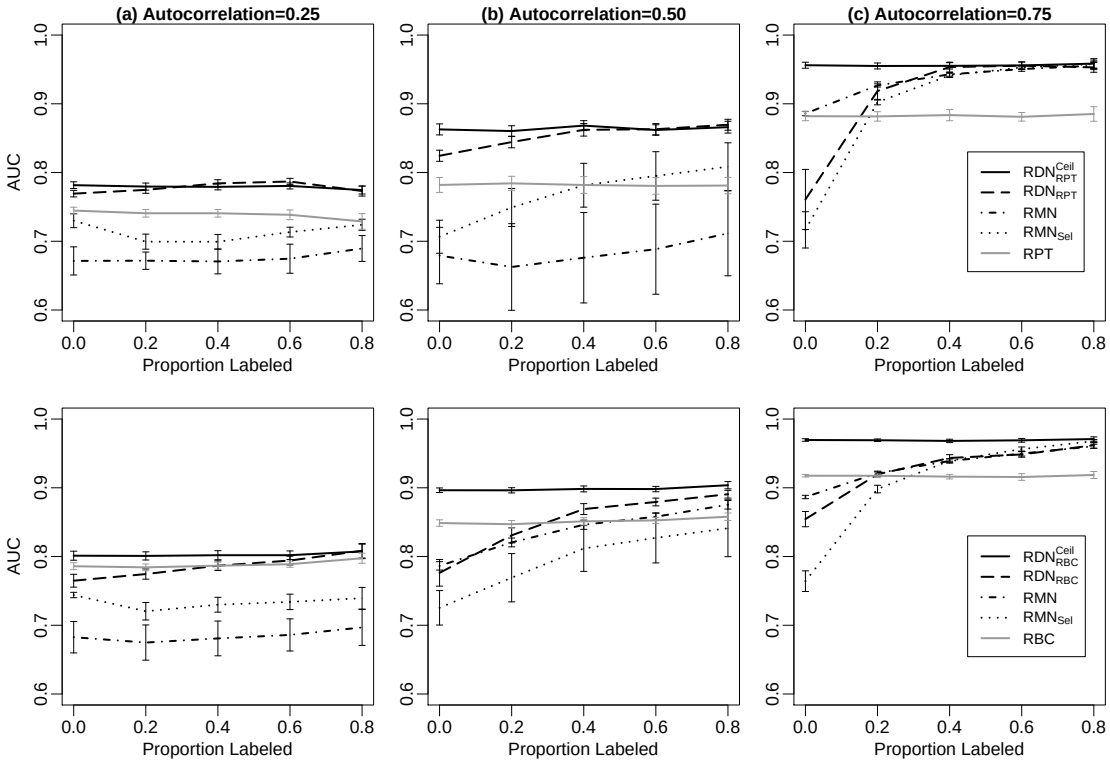


Figure 7: Evaluation of RDN inference.

probabilities will be biased. When there is high autocorrelation, $RDN_{RPT}$ performance is indistinguishable from $RMN$, except when there are no labels to seed inference—the same situation where $RDN_{RPT}$ fails to meet its ceiling. When autocorrelation is high, the mismatch in feature space is not a problem. In this situation most neighbors share similar attribute values, thus the RMN features are able to accurately capture the data dependencies.

The data for the second set of experiments (bottom row) were generated with an $RDN^*_{RBC}$. The $RDN_{RBC}$ feature space is roughly comparable to the RMN because the $RDN_{RBC}$ uses multinomials to model individual neighbor attribute values. On these data, $RDN_{RBC}$ performance is superior to $RMN$ performance only when there is low autocorrelation. $RMN_{Sel}$ uses fewer features than $RMN$ and it has superior performance on the data with low autocorrelation, indicating that the RMN learning algorithm may be overfitting the feature weights and producing biased probability estimates. We experimented with a range of priors to limit the impact of weight overfitting, but the effect remained consistent.

$RDN_{RBC}$ performance is superior to $RBC$ performance only when there is moderate to high autocorrelation and sufficient seed information. When autocorrelation is low, the RBC is comparable to both the $RDN^{ceil}_{RBC}$ and the $RDN_{RBC}$. Even when autocorrelation is moderate or high, RBC performance is still relatively high. Since the RBC is low-variance and there are only four attributes in our datasets, it is not surprising that the RBC 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.

To investigate Gibbs sampling convergence, we tracked AUC throughout the RDN Gibbs sampling procedure. Figure 8 demonstrates AUC convergence on each inference task described above. We selected a single learned model at random from each task and report convergence from the trials corresponding to five different test sets. AUC improves very quickly, often leveling off within the first 250 iterations. This shows that the approximate inference techniques employed by the RDN may be quite efficient to use in practice. However, when autocorrelation is high, longer chains may be necessary to ensure convergence. There are only two chains that show a substantial increase in performance after 500 iterations and both occur in highly autocorrelated datasets. Also, the $RDN_{RBC}$ chains exhibit significantly more variance than the $RDN_{RPT}$ chains, particularly when autocorrelation is high. This may indicate that the use of longer Gibbs chains, or an approach that averages predictions obtained from multiple random restarts, would improve performance.

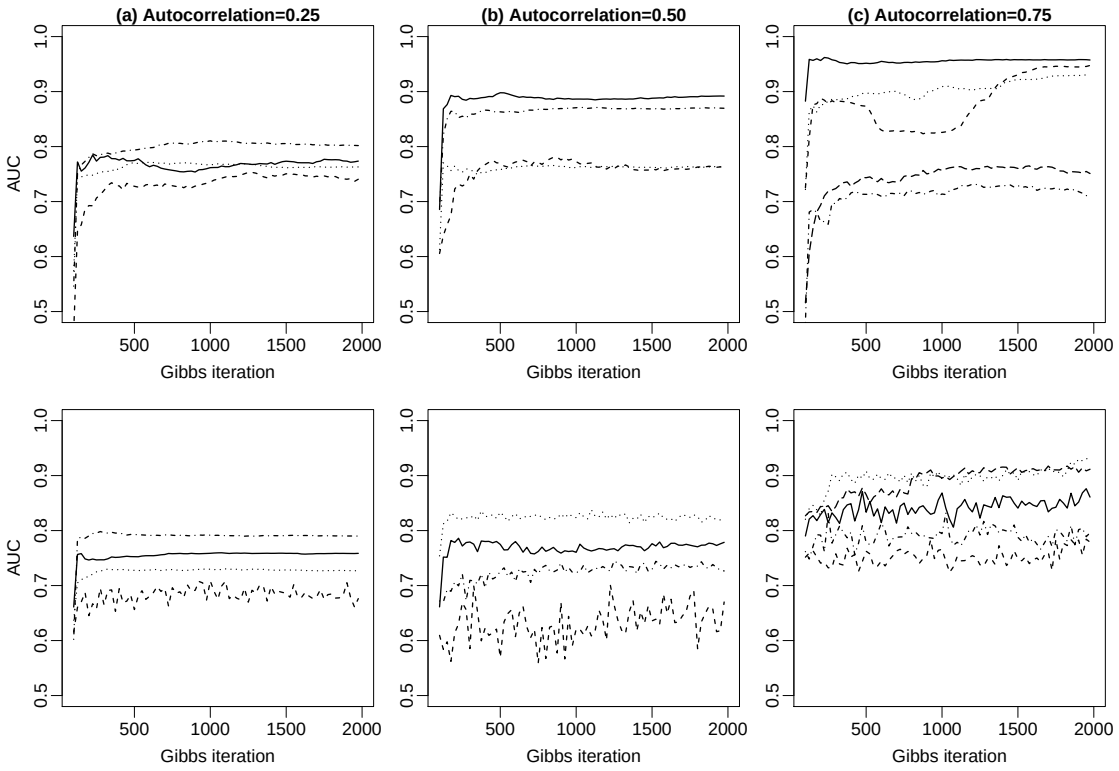


Figure 8: Gibbs convergence rates for five different trials of $RDN_{RPT}$ (top row) and $RDN_{RBC}$ (bottom row).

### 4.2 Empirical Data Experiments

We learned RDNs for five real-world relational datasets. Figure 9 depicts the objects and relations in each dataset. Section 4.2.1 illustrates the types of domain knowledge that can be learned with the selective $RDN_{RPT}$. Section 4.2.2 evaluates both the $RDN_{RPT}$ and the $RDN_{RBC}$ in a prediction context, where the values of a single attribute are unobserved.

The first 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 second data set (Gene) is a relational data set containing information about the yeast genome at the gene and the protein level.[19] The data set contains information about 1,243 genes and 1,734 interactions among their associated proteins.

The third dataset is drawn from the Internet Movie Database (IMDb).[20] 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.

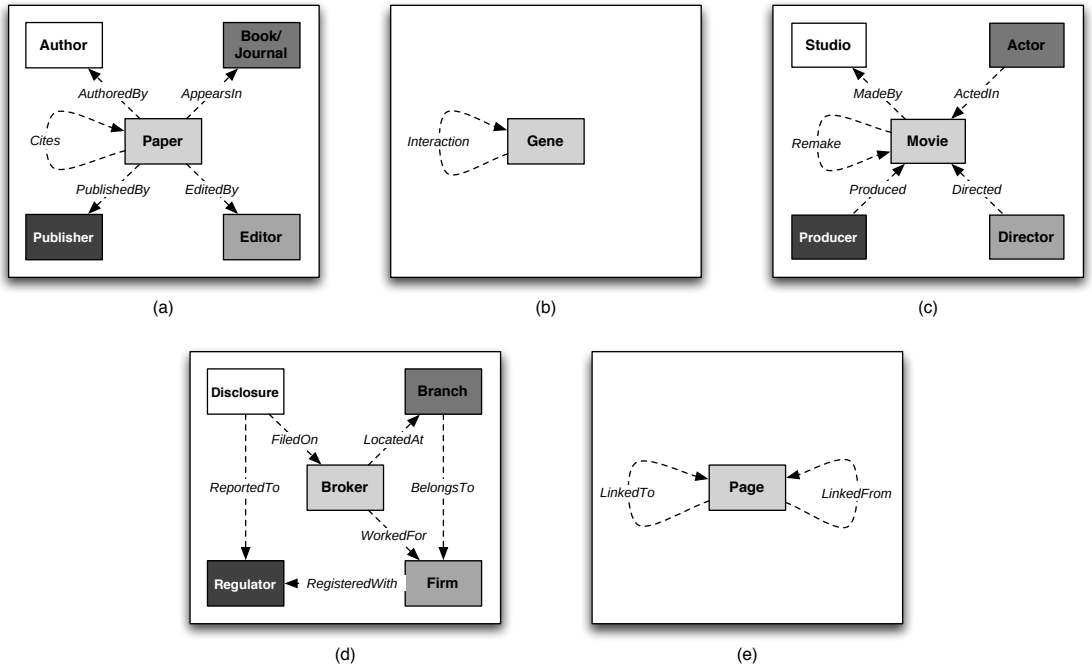


Figure 9: Data schemas for (a) Cora, (b) Gene, (c) IMDb, (d) NASD, and (e) WebkKB.

The fourth dataset is from the National Association of Securities Dealers (NASD) (Neville et al., 2005). It is drawn from NASD's Central Registration Depository (CRD©)

19. See http://www.cs.wisc.edu/∼dpage/kddcup2001/.
20. See http://www.imdb.com.

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 approximately 10,000 brokers who were active in the years 1997-2001, along with 12,000 associated branches, firms, and disclosures.

The fifth data set was collected by the WebKB Project (Craven et al., 1998). The data comprise 3,877 web pages from four computer science departments. The web pages have been manually labeled with the categories course, faculty, staff, student, research project, or other. The collection contains approximately 8,000 hyperlinks among the pages.

#### 4.2.1 RDN Models

The RDNs in Figures 10-14 continue with the RDN representation introduced in Figure 2b. Each item type is represented by a separate plate. An arc from $x$ to $y$ indicates the presence of one or more features of $x$ in the conditional model learned for $y$. Arcs inside a plate represent dependencies among the attributes of a single object. Arcs crossing plate boundaries represent dependencies among attributes of related objects, with edge labels indicating the underlying relations. 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 10 paper topic is influenced by the topics of other papers written by the paper's authors, so the arc is labeled with two *AuthoredBy* relations and a small *A* rectangle indicating an *Author* object.

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 10 author rank is influenced by the number of paper written by the author.

For each dataset, we learned an $RDN_{RPT}$ with queries that included all neighbors up to two links away in the data graph. For example in Cora, when learning an RPT of a paper attribute, we considered the attributes of associated authors and journals, as well as papers related to those objects.

On the Cora data, we learned an RDN for seven attributes. Author rank records ordering in paper authorship (e.g., first author, second author). Paper type records category information (e.g., PhD thesis, technical report); topic records content information (e.g., genetic algorithms, reinforcement learning); year and month record publication dates. Journal name-prefix records the first four title letters (e.g., IEEE, SIAM); book-role records type information (e.g., workshop, conference).

Figure 10 shows the resulting RDN. The RDN learning algorithm selected 12 of the 139 dependencies considered for inclusion in the model. Four of the attributes—author rank, paper topic, paper type, and paper year—exhibit autocorrelation dependencies. 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.

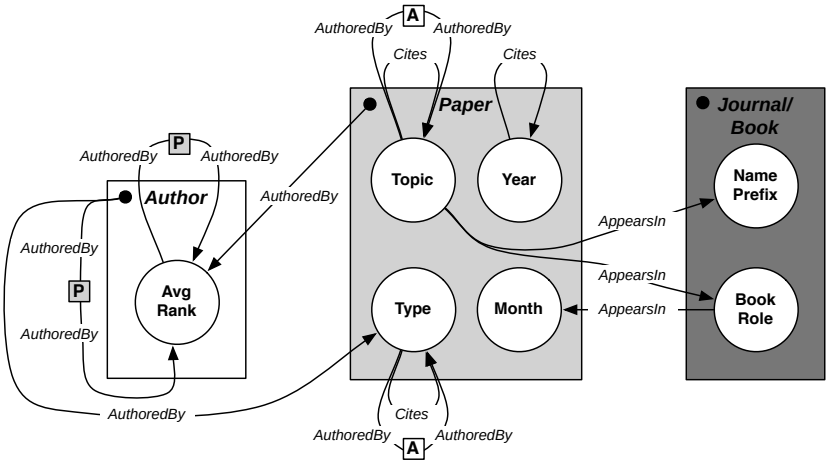


Figure 10: RDN for the Cora dataset.

Exploiting these types of autocorrelation dependencies 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.

The Gene data contain attributes associated with both objects and links (i.e., interactions). We learned an RDN for seven attributes. Gene function records activities of the proteins encoded by the genes; location records each protein's localization in the cell; phenotype records characteristics of individuals with a mutation in the gene/protein; class records the protein type (e.g., transcription factor, protease); essential records whether the gene is crucial to an organism's survival. Interaction expression records the correlation between gene expression patterns for pairs of interacting genes; type records interaction characteristics (e.g., genetic, physical).

Figure 11 shows the resulting RDN. The RDN learning algorithm selected 19 of the 77 dependencies considered for inclusion in the model. In these data, all the gene attributes exhibit autocorrelation—this is strong evidence that there are regularities among the genes whose proteins interact in the cell.

On the IMDb data, we learned an RDN for ten discrete attributes. First-movie-year records the date of the first movie made by a director or studio; has-award records whether a director or actor has won an Academy award; in-US records whether a studio is located in the US; receipts records whether a movie made more than $2 million in the opening weekend box office; genre records a movie's type (e.g., drama, comedy); hsx-rating records an actor's value on the Hollywood Stock Exchange (www.hsx.com); birth-year and gender record demographic information.

Figure 12 shows the resulting RDN. The RDN learning algorithm selected 29 of the 170 dependencies considered for inclusion in the model. Again we see that four of the attributes

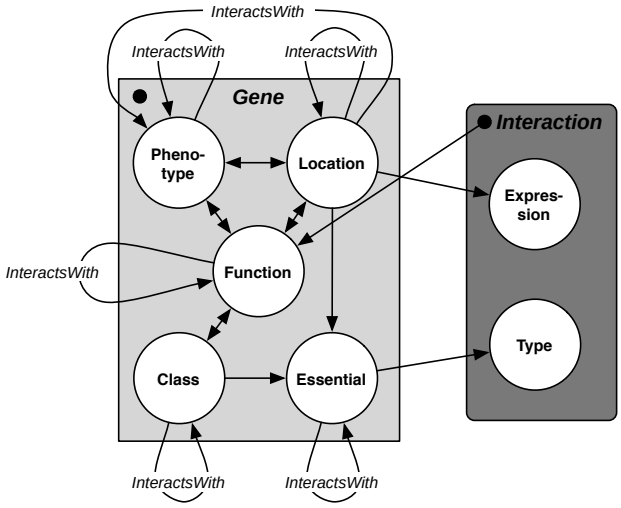


Figure 11: RDN for the Gene dataset.

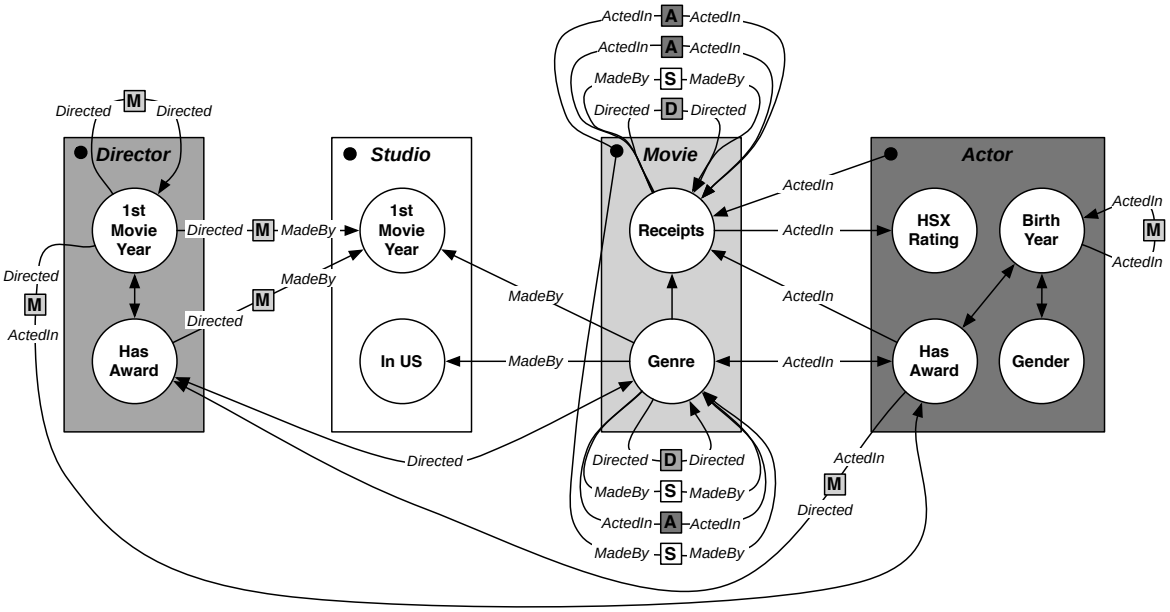


Figure 12: RDN for the IMDb dataset.

exhibit autocorrelation. Movie receipts and genre each exhibit a number of autocorrelation dependencies (through actors, directors, and studios), which illustrates the group structure of the Hollywood movie industry.

On the NASD data, we learned an RDN for eleven attributes. Firm size records the number of employed stockbrokers each year; layoffs records the number of terminations each year. On-watchlist records whether a firm or broker is under heightened supervision. Broker is-problem and problem-in-past record whether a broker is, or has been, involved in serious misconduct; has-business records whether a broker owns a business on the side. Disclosure type and year record category (e.g., customer complaint) and date information regarding

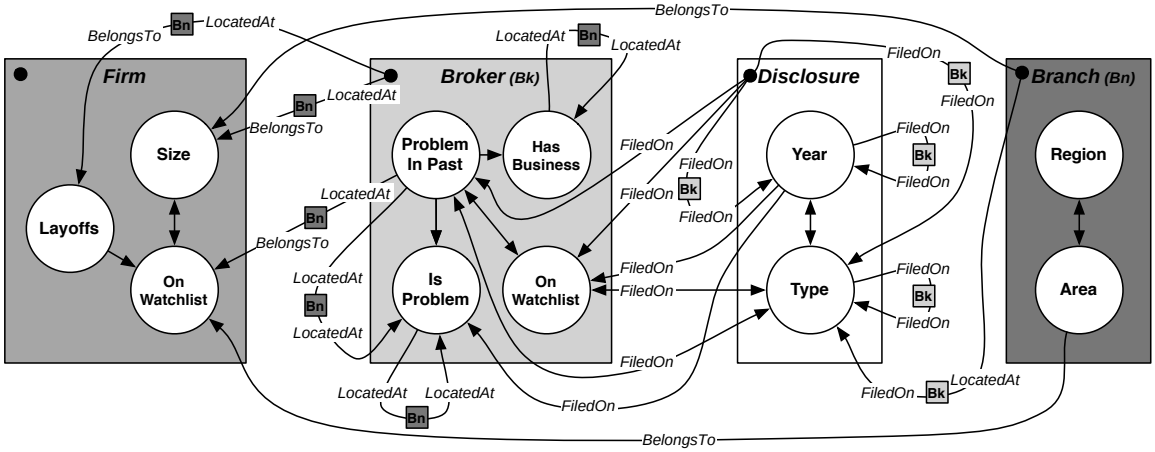


Figure 13: RDN for the NASD dataset (1999).

disciplinary events (filed on brokers). Region and area record location information about branches.

Figure 13 shows the resulting RDN. The RDN learning algorithm selected 32 of the 160 dependencies considered for inclusion in the model. Again we see that four of the attributes exhibit autocorrelation. Subjective inspection by NASD analysts indicates that the RDN had 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—experts are more likely to trust, and make regular use of, models they can understand.

On the WebKB data, we learned an RDN for four attributes. School records page location (e.g., Cornell, Washington); label records page type (e.g., student, course); URL-server records the first portion of the server name following *www* (e.g., cs, engr); URL-text records the name of the first directory in the URL path (e.g., UTCS, department). Figure 14 shows the resulting RDN. The RDN learning algorithm selected 9 of the 52 dependencies considered for inclusion in the model. All of the attributes exhibit autocorrelation in the WebKB data.

### 4.2.2 Prediction

We evaluated RDNs on prediction tasks in order to assess (1) whether autocorrelation dependencies among instances can be used to improve model accuracy, and (2) whether the RDNs, using Gibbs sampling, can effectively infer labels for a network of instances. To do this, we compared the same four classes of models used in Section 4.1: conditional models, RDNs, ceiling RDNs, and RMNs.

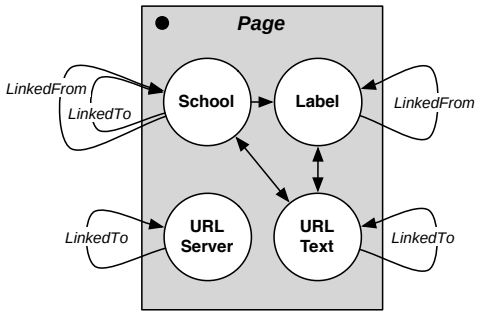


Figure 14: RDN for the WebKB dataset.

We used the following prediction tasks: movie receipts for IMDb, paper topic for Cora, page label for WebKB, gene location for Gene, and broker is-problem for NASD. For each dataset, we used queries that included all neighbors up to two links away in the data graph to learn $RDN_{RBC}$ and $RDN_{RPT}$ models. Recall that the $RDN_{RBC}$ uses the entire set of attributes in the resulting subgraphs and the $RDN_{RPT}$ performs feature selection over the attribute set.

Figure 15 shows AUC results for the first three model types on the five prediction tasks. (We discuss RMN results below.) Figure 15a graphs the results of the $RDN_{RPT}$, compared to the $RPT$ conditional model. Figure 15b graphs the results of the $RDN_{RBC}$, compared to the $RBC$ conditional model.

The graphs show AUC for the most prevalent class, averaged over a number of training/test splits. For Cora, IMDb, and NASD, we used temporal sampling where we learned models on one year of data and applied the models to the subsequent year. There were four temporal samples for IMDb and NASD, and five for Cora. For WebKB we used cross-validation by department, learning on three departments and testing on pages from the fourth, held-out department. For Gene there was no clear sampling choice, so we used ten-fold cross validation on random samples of genes. When there were links between the test and training sets, the class labels of the training set were made available to the RDNs and RMNs for use during inference. 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 the conditional and ceiling models, with a null hypothesis of no difference in the AUC.

When using the RPT as the conditional learner (Figure 15a), RDN performance is superior to RPT performance on all tasks. The difference is statistically significant for three of the five tasks. This indicates that autocorrelation is both present in the data and identified by the RDNs. As mentioned previously, the RPT can sometimes use the observed attributes of related items to effectively reason with autocorrelation dependencies. However, in some cases the observed attributes 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 10). On all tasks, the RDNs achieve comparable performance to the ceiling models. This indicates that the RDN achieved the same level of performance as if it had access to the true labels of related objects. 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

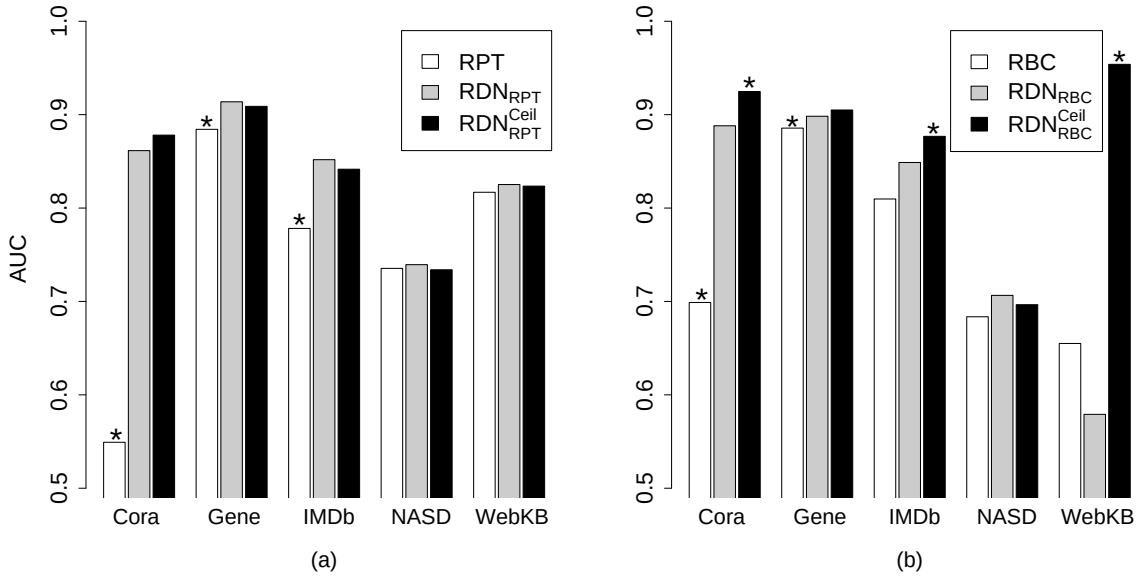


Figure 15: AUC results for (a) $RDN_{RPT}$ and RPT, and (b) $RDN_{RBC}$ and RBC. Asterisks denote model performance that is significantly different ($p < 0.10$) from $RDN_{RPT}$ and $RDN_{RBC}$.

objects. Indeed, on a number of the datasets, RDN performance is slightly higher than that of the ceiling model.

Similarly, when using the RBC as the conditional learner (Figure 15b), the performance of RDNs is superior to the RBCs on all but one task and statistically significant for two of the tasks. Notably, on the WebKB data RDN performance is worse than that of the RBC. However, the ceiling performance is significantly higher than RBC. This indicates that autocorrelation dependencies are identified by the RDN but the model is unable to exploit those dependencies during Gibbs sampling. This effect is due to the amount of information available to seed the inference process. There is sparse information in the attributes other than page label, and because the departments are nearly disjoint, there are few labeled instances before inference. This leaves the RDN with little information to anchor its predictions, which results in marginal predictions closer to random. Similar behavior appeared in the synthetic data experiments, indicating that the $RDN_{RBC}$ may need more information to seed the inference process.

The $RDN_{RBC}$ achieves comparable performance to the ceiling models on only two of the five tasks. This may be another indication that RDNs 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 $RDN_{RBC}$ does not significantly outperform the $RDN_{RPT}$ on any of the tasks, the $RDN^{Ceil}_{RBC}$ is significantly higher than $RDN^{Ceil}_{RPT}$ for Cora and IMDb. This indicates that, when there is enough seed information, the $RDN_{RBC}$ may achieve significant performance gains over the $RDN_{RPT}$.

| | **Num $RDN$ attributes** | **Num selected $RDN$ attributes** | **Num $RMN$ features** | **Num $RMN_{Sel}$ features** |
|---|---|---|---|---|
| Cora | 22 | 2.0 | 1029 | 98.0 |
| Gene | 19 | 3.1 | 3640 | 606.0 |
| IMDb | 15 | 5.0 | 234 | 90.5 |
| NASD | 16 | 5.7 | 526 | 341.5 |
| WebKB | 34 | 5.0 | 2478 | 270.0 |

Table 3: Number of attributes/features used by RDNs and RMNs.

Due to time and memory constraints, we were unable to learn RMNs for all but two of the real-data experiments. Table 3 reports information for each dataset—the number of attributes available to the RDNs, the average number of attributes selected by the RDNs, and the number of features constructed by the $RMN$ and the $RMN_{Sel}$. Recall that the $RMN$ constructs its features from all the attributes available to the RDN and the $RMN_{Sel}$ constructs its features from the attributes selected by the RDN. Due to the size of the feature spaces considered by the non-selective $RMN$, we were unable learn models for any of the datasets. We were able to successfully learn an $RMN_{Sel}$ for the Cora and IMDb datasets. The average AUC of the $RMN_{Sel}$ was 74.4% for Cora and 60.9% for IMDb. This is far below the RDN results reported in Figure 15. Note that previous RMN results (Taskar et al., 2002) report accuracy of the most likely labeling for the entire dataset. In contrast, we are evaluating AUC of the marginal probabilities for each instance (inferred jointly). This may indicate that RMN inference will produce biased (marginal) probability estimates when run in a "loopy" relational network, due to overfitting the clique weights. When skewed weights are applied to collectively infer the labels throughout the test set, the inference process may converge to extreme labelings (e.g., all positive labels in some regions of the graph, all negative labels in other regions), which would bias probability estimates for many of the instances.

## 5. Related Work

There are three types of statistical relational models relevant to RDNs: probabilistic relational models, probabilistic logic models, and collective inference models. We discuss each of these below.

### 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 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), 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.

An RBN for $\mathbf{X}$ uses a directed model graph $G_M = (V_M, E_M)$ and a set of conditional probability distributions $P$ to represent a joint distribution over $\mathbf{X}$. Each node $v \in V_M$ corresponds to an $X_k^t \in \mathbf{X}$. The set $P$ contains a conditional probability distribution for each variable given its parents, $p(x_k^t | pa_{x_k^t})$. Given $(G_M, P)$, the joint probability for a set of values $\mathbf{x}$ is computed as a product over the item types $T$, the attributes of that type $X^t$, and the items of that type $v, e$:

$$p(\mathbf{x}) = \prod_{t \in T} \prod_{X_i^t \in X^t} \prod_{v:T(v)=t} p(x_{vi}^t | pa_{x_{vi}^t}) \prod_{e:T(e)=t} p(x_{ei}^t | pa_{x_{ei}^t})$$

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 mentioned in Section 1, the acyclicity requirement precludes the learning of arbitrary autocorrelation dependencies and limits the applicability of these models in relational domains. The PRM representation, which ties parameters across items of the same type, makes it difficult for RBNs to represent autocorrelation dependencies. Because autocorrelation is a dependency among the value of the same variable on linked items of the same type, the CPDs that represent autocorrelation will produce cycles in the rolled out inference graph. For example, when modeling the autocorrelation among the topics of two hyperlinked web pages $P_1$ and $P_2$, the CPD for the topic of $P_1$ will have the topic of $P_2$ as a parent and vice versa. In general, unless there is causal knowledge of how to structure the order of autocorrelation dependencies (e.g., parents' genes will never be influenced by the genes of their children), it is impossible to tie the parameters and still guarantee an acyclic graph after roll out. Note that this is not a problem for undirected models such as RMNs or RDNs. Thus, RDNs enjoy the strengths of RBNs (namely, understandable knowledge representation and efficient learning) without being constrained by an acyclicity requirement.

An RMN for $\mathbf{X}$ uses a undirected model graph $U_M = (V_M, E_M)$ and a set of potential functions $\Phi$ to represent a joint distribution over $\mathbf{X}$. Again each node $v \in V_M$ corresponds to an $X_k^t \in \mathbf{X}$. RMNs use relational clique templates $CT$ to specify the ways in which cliques are defined in $U_M$. Cliques templates are defined over a set of item types, a boolean constraint on how the types must relate to each other in the data graph, and a set of

attributes to consider on the matching items. Let $C(CT)$ be the set of cliques in the graph $U_M$ that match a specific clique template $C \in CT$. As with Markov networks, each clique $c \in C(CT)$ is associated with a set of variables $X_c$ and a clique potential $\phi_c(x_c)$. Given $(U_M, \Phi)$, the joint probability for a set of values $\mathbf{x}$ is computed as a product over the clique templates in $CT$ and the matches to the clique template $c$:

$$p(\mathbf{x}) = \frac{1}{Z} \prod_{C_i \in CT} \prod_{c_j \in C(C_i)} \phi_{c_j}(x_{c_j})$$

The RMN learning algorithm (Taskar et al., 2002) uses maximum-a-posteriori parameter estimation with Gaussian priors, modifying 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, RDNs 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 fields (McCallum, 2003) and Markov logic networks (Kok and Domingos, 2005).

### 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, 2006).

PLMs represent a joint probability distribution over the ground atoms of a first-order knowledge base. The first-order knowledge base contains a set of first-order formulae, and the PLM associates a set of weights/probabilities with each of the formulae. Combined with a set of constants representing objects in the domain, PLMs specify a probability distribution over possible truth assignments to ground atoms 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 (MLNs) 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. An MLN, unrolled over a set of objects in the domain, specifies an undirected Markov network. In this sense, they share the same strengths and weaknesses as RMNs—they are capable of representing cyclic autocorrelation relationships but suffer from decreased efficiency if full joint estimation is used during learning.

Recent work on structure learning for MLNs (Kok and Domingos, 2005) has investigated the use of pseudolikelihood to increase learning efficiency. The MLN structure learning algorithm restricts the search space by limiting the number of distinct variables in a clause, and then uses beam search, with a weighted pseudolikelihood scoring function, to find the best clauses to add to the network.

Although both RDNs and MLNs use pseudolikelihood techniques to learn model structures efficiently, they each employ a different representational formalism. In particular, MLNs use weighted logic formulae while RDNs use aggregate features in CPDs. Our future work will investigate the performance tradeoffs between RDN and MLN representations when combined with pseudolikelihood estimation.

### 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 models (Macskassy and Provost, 2003, 2004). These approaches to collective inference were developed in an ad hoc procedural fashion, motivated by the observation that they appear to work well in practice. RDNs formalize this approach in a principled framework—learning models locally (maximizing pseudolikelihood) 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.

## 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 RDNs is the ability to efficiently learn and reason with autocorrelation. Autocorrelation is a nearly ubiquitous phenomenon in relational datasets and the resulting dependencies are often cyclic in nature. If a dataset exhibits autocorrelation, an RDN can learn the associated dependencies and then 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. Furthermore, our experiments show that inference with RDNs is comparable, or superior, to RMN inference over a range of conditions, which indicates that pseudolikelihood estimation can be used effectively to learn an approximation of the full joint distribution.

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 RDNs to Markov logic networks in order to evaluate the performance tradeoffs for using pseudolikelihood approximations with different relational representations. Also, because our analysis indicates that each model reacts differently to the amount of seed information and level of autocorrelation, future work will attempt to quantify these effects on performance more formally. In particular, we are developing a bias/variance framework to decompose model errors into components of both the learning and inference processes (Neville and Jensen, 2006). Conventional bias/variance analysis is a useful tool for investigating the performance of machine learning algorithms, but it decomposes loss into errors due to aspects of the learning process alone. In relational and network applications, the collective inference process introduces an additional source of error—both through the use of approximate inference algorithms and through variation in the availability of test set information. Our framework can be used to evaluate hypotheses regarding the mechanisms behind poor model performance (e.g., identifiability problems increase RDN variance) and investigate algorithmic modifications designed to improve model performance.

## Acknowledgments

The authors acknowledge the invaluable assistance of A. Shapira, as well as helpful comments from C. Loiselle and two anonymous reviewers.

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.

## Appendix A. Synthetic Data Generation Models

We detail the manually specified conditional models used in Section 4.1. Specifically, there are three RBC and three RPT models. Each one is designed to generate data with low, medium, or high levels of autocorrelation (0.25, 0.50, 0.75).

$$
\begin{aligned}
RBC := \quad & p(X_1|\mathbf{X_1}_R, X_2) \propto \prod_R p(X_{1R}|X_1) \cdot p(X_2|X_1) \\
RBC_{0.25}: \quad & p(X_{1R}|X_1) := p(X_{1R}{=}1|X_1{=}1) = p(X_{1R}{=}0|X_1{=}0) = 0.5625 \\
& p(X_2|X_1) := p(X_2{=}1|X_1{=}1) = p(X_2{=}0|X_1{=}0) = 0.75 \\
RBC_{0.50}: \quad & p(X_{1R}|X_1) := p(X_{1R}{=}1|X_1{=}1) = p(X_{1R}{=}0|X_1{=}0) = 0.6250 \\
& p(X_2|X_1) := p(X_2{=}1|X_1{=}1) = p(X_2{=}0|X_1{=}0) = 0.75 \\
RBC_{0.75}: \quad & p(X_{1R}|X_1) := p(X_{1R}{=}1|X_1{=}1) = p(X_{1R}{=}0|X_1{=}0) = 0.6875 \\
& p(X_2|X_1) := p(X_2{=}1|X_1{=}1) = p(X_2{=}0|X_1{=}0) = 0.75
\end{aligned}
$$

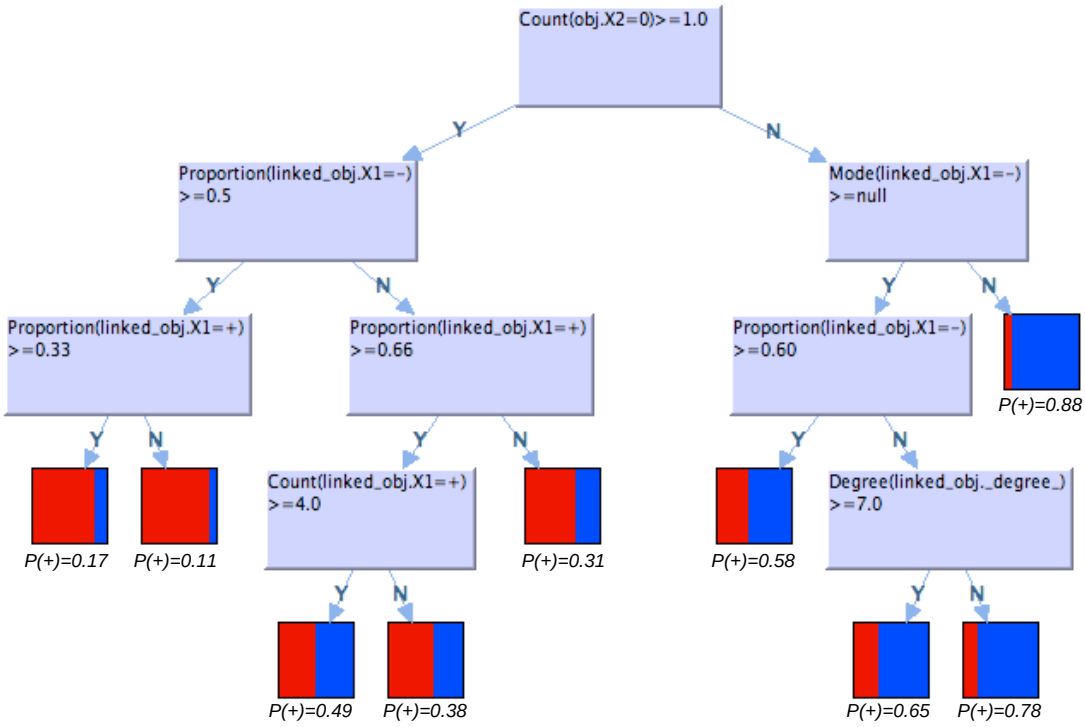


Figure 16: $RPT_{0.25}$ model used for synthetic data generation with low autocorrelation.

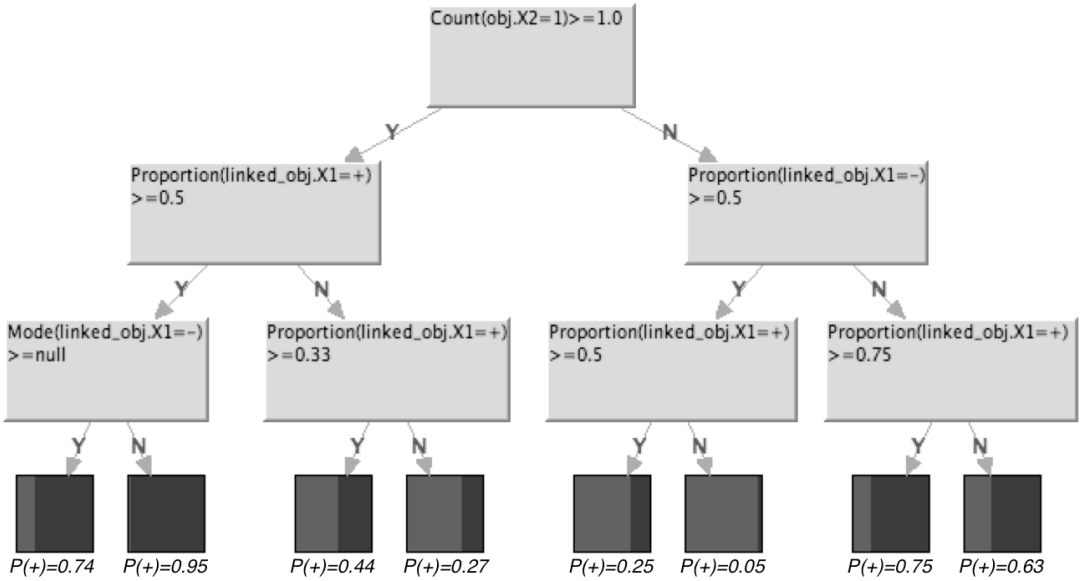


Figure 17: $RPT_{0.50}$ model used for synthetic data generation with medium autocorrelation.

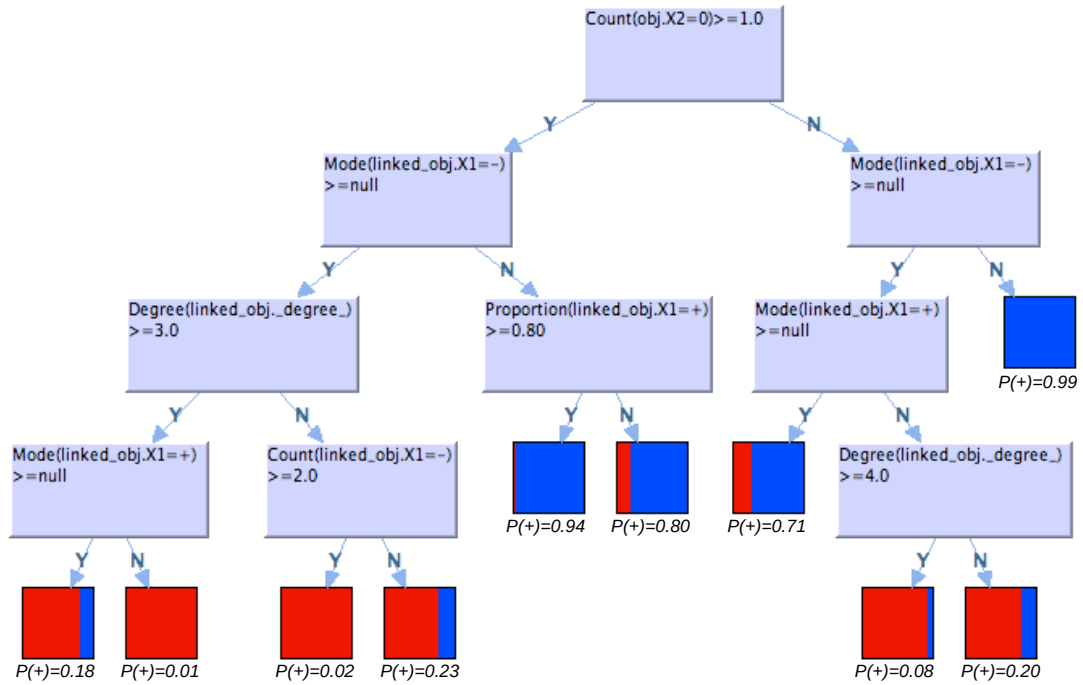


Figure 18: $RPT_{0.75}$ model used for synthetic data generation with high autocorrelation levels.

## 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. Spatial interaction and the statistical analysis of lattice systems. *Journal of the Royal Statistical Society, Series B*, 36(2):192–236, 1974.

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.

G. Casella and R. Berger. *Statistical Inference*. Duxbury, 2002.

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.

F. Comets. On consistency of a class of estimators for exponential families of Markov random fields on the lattice. *The Annals of Statistics*, 20(1):455–468, 1992.

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.

M. Craven, D. DiPasquo, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and S. Slattery. Learning to extract symbolic knowledge from the World Wide Web. In *Proceedings of the 15th National Conference on Artificial Intelligence*, pages 509–516, 1998.

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.

T. Fawcett and F. Provost. Adaptive fraud detection. *Data Mining and Knowledge Discovery*, 1(3):291–316, 1997.

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.

B. Gidas. Consistency of maximum likelihood and pseudolikelihood estimators for Gibbs distributions. In *Proceedings of the Workshop on Stochastic Differential Systems with Applications in Electrical/Computer Engineering, Control Theory, and Operations Research*, pages 129–145, 1986.

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.

S. Hill, F. Provost, and C. Volinsky. Network-based marketing: Identifying likely adopters via consumer networks. *Statistical Science*, 22(2), 2006.

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.

J. Lafferty, A. McCallum, and F. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the 18th International Conference on Machine Learning*, pages 282–289, 2001.

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

E. Lehmann and G. Casella. *Theory of Point Estimation*. Springer-Verlag, New York, 1998.

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 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. Dependency networks for relational data. In *Proceedings of the 4th IEEE International Conference on Data Mining*, pages 170–177, 2004.

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 and D. Jensen. Iterative classification in relational data. In *Proceedings of the Workshop on Statistical Relational Learning, 17th National Conference on Artificial Intelligence*, pages 42–49, 2000.

J. Neville and D. Jensen. Bias/variance analysis for network data. In *Proceedings of the Workshop on Statistical Relational Learning, 23rd International Conference on Machine Learning*, 2006.

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.

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.

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. *Machine Learning*, 62:107–136, 2006.

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, 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. 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.

H. White. *Estimation, Inference and Specification Analysis*. Cambridge University Press, New York, 1994.

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.