Collective Classification with Relational Dependency Networks

Jennifer Neville and David Jensen

Department of Computer Science
140 Governors Drive
University of Massachusetts, Amherst
Amherst, MA 01003
{jneville|jensen}@cs.umass.edu

Collective classification models exploit the dependencies in a network of objects to improve predictions. For example, in a network of web pages, the topic of a page may depend on the topics of hyperlinked pages. A relational model capable of expressing and reasoning with such dependencies should achieve superior performance to relational models that ignore such dependencies. In this paper, we present relational dependency networks (RDNs), extending recent work in dependency networks to a relational setting. RDNs are a collective classification model that offers simple parameter estimation and efficient structure learning. On two real-world data sets, we compare RDNs to ordinary classification with relational probability trees and show that collective classification improves performance.

1 Introduction

In this paper, we show how *autocorrelation* can be used to improve the accuracy of statistical models of relational data. Autocorrelation is a statistical dependency between the values of the same variable on related entities and is a common characteristic of many relational data sets. In previous work, we have shown how autocorrelation can cause bias in learning algorithms (Jensen & Neville 2002), particularly when it appears with *concentrated linkage*, another common characteristic of relational data sets. However, this work explores methods which exploit autocorrelation to improve model accuracy. In particular, we demonstrate this can be accomplished with a relatively simple approach to structure learning in a class of undirected graphical models that we call relational dependency networks (RDNs).

It is relatively easy to understand how autocorrelation could be used to improve the predictions of statistical models. For example, consider the problem of automatically predicting the *topic* of a technical paper (e.g., neural networks, reinforcement learning, genetic algorithms). One simple method for predicting paper topics would look at the position of a paper within a citation graph. It may be possible to predict a given paper's topic with high accuracy based on the topics of neighboring papers in this graph. This is possible only because we expect high autocorrelation in the citation graph—papers tend to cite other papers with the same topic.

However, such a scheme for prediction assumes that all the labels of related entities (e.g., topics of referenced papers) are known. In many cases, the topics of an entire set of papers may need to be inferred simultaneously. This approach, called *collective classification* (Taskar, Abbeel & Koller 2002) requires both models and inference procedures that can use inferences about one entity in a relational data set to influence inferences about related entities. Similar approaches have been used in several recent applications (Neville & Jensen 2000; Chakrabarti, Dom & Indyk 1998).

In this paper, we introduce relational dependency networks (RDNs), an undirected graphical model for relational data. We show how RDNs can be learned and how RDNs and Gibbs sampling can be used for collective classification. Because they are undirected graphical models, RDNs can represent the cyclic dependencies required to express autocorrelation, and they can express a joint probability distribution, rather than only a single conditional distribution. In addition, they are relatively simple to learn and easy to understand.

We show preliminary results indicating that collective inference with RDNs offers improved performance over non-collective inference that we term "individual inference." We also show that RDNs applied collectively can perform near the theoretical ceiling achieved if all labels of neighbors are known with perfect accuracy. These results are very promising, indicating the potential utility of additional exploration of collective inference with RDNs.

2 Classification Models of Relational Data

Many relational models are used for "individual inference," where inferences about one instance are not used to change the inference of related instances. For example, some work in inductive logic programming (ILP) and relational learning uses relational instances that can be represented as disconnected subgraphs (e.g., molecules), thus removing the need (and the opportunity) for collective inference. Such models can cope with complex relational structure *of an instance*, but they do not attempt to model the relational structure *among instances*.

Even learning relational models for individual inference can be difficult because learning in relational data differs substantially from learning in propositional settings. For example, work in ILP has long considered difficult representational issues such as recursion, aggregation, variables, quantification, and other aspects of first-order and higher-order logics. In addition, some of our recent work has examined the unique statistical properties of relational data, and the influence of these characteristics on learning algorithms. We have shown how concentrated linkage and relational autocorrelation can bias relational learners toward particular features (Jensen & Neville 2002). We have also shown how degree disparity can cause misleading correlations in relational data, leading learning algorithms to add excessive structure to learned models (Jensen, Neville & Hay 2003).

To address the challenges of relational learning, new learning algorithms are necessary. For example, probabilistic relational models (PRMs) (Getoor, Friedman, Koller & Pfeffer 2001) are a form of directed graphical model thatextend Bayesian networks to support reasoning in complex relational domains. Unfortunately, PRMs are a directed model in which autocorrelation cannot be represented due to acyclicity constraints.[1]

As another example, we have recently developed relational probability trees (RPTs), a method for learning tree-structured models that encode conditional probability distributions for a class label that depend on both the attributes of related instances and the features of the surrounding structure (Neville, Jensen, Friedland & Hay 2003). Our methods for learning RPTs adjust for the sources of bias mentioned above. As a result, the learned RPTs are a relatively compact and parsimonious representation of conditional probability distributions in relational data. Until recently, we had only applied these models for individual inference. The algorithm for learning RPTs adjusts for statistical biases caused by autocorrelation, but our inference techniques have not made use of autocorrelation to improve inference.

This is unfortunate, because autocorrelation is a nearly ubiquitous phenomenon in relational data. We have observed relatively high levels of autocorrelation in relational data sets. For example, in analysis of the 2001 KDD Cup data we found that the proteins located in the same place in a cell (e.g., mitochondria or cell wall) had highly autocorrelated functions (e.g., transcription or cell growth). Autocorrelation has been identified by other investigators as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes, Pregibon & Volinsky 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 2001).

We also know that exploiting autocorrelation can result in significant increases in predictive accuracy. Several recent developments in relational learning have focused on exploiting autocorrelation. For example, the relational vector-space (RVS) model (Bernstein, Clearwater & Provost 2003) uses weighted adjacency vectors to construct classifiers. The model is extremely simple, but it produces accurate classifiers in data with strong autocorrelation. Other examples include work with web pages (Chakrabarti et al. 1998) that uses the hyperlink structure to produce smoothed estimates of class labels and our own prior work (Neville & Jensen 2000) that uses an iterative classification scheme to improve accuracy by exploiting the inferred class labels of related instances.

Recently, undirected graphical models capable of representing and reasoning with autocorrelation have been explored for the task of modeling relational data. Domingos and Richardson (2001) represent market entities as social networks and develop Markov random field models to model the influence in purchasing patterns throughout the network. Taskar et al. (2002) use relational Markov networks (RMNs), based on conditional random fields for sequence data (Lafferty, McCallum & Pereira 2001), to model the dependencies among web pages to predict page type. Undirected models have proved to be successful for collective classification of relational data. However, research into these models has focused primarily on parameter estimation and inference procedures. Model structure is not learned automatically, it is pre-specified by the user. Also, the models do not automatically identify which features are most relevant to the task. For relational tasks, which are likely to have a large

[1] An exception is where autocorrelation is structured by some additional information such as temporal constraints. (Getoor et al. 2001)

number of features, this lack of selectivity will make the model more difficult to interpret and understand.

3 Relational Dependency Networks

Relational dependency networks extend recent work on dependency networks (Heckerman, Chickering, Meek, Rounthwaite, and Kadie 2000) to a relational setting. Dependency networks (DNs) are graphical models of probabilistic relationships—an alternative to Bayesian networks and Markov random fields. DNs are easy to learn and have been shown to perform well for a number of propositional tasks so we expect them to offer similar advantages when used in a relational setting. We begin by reviewing the details of dependency networks for propositional data and then describe how to extend dependency networks for use with relational data.

3.1 Dependency Networks

Like Bayesian networks and Markov random fields, dependency networks encode probabilistic relationships among a set of variables. Dependency networks are an alternative form of graphical model that approximate the full joint distribution with a set of conditional distributions that are learned independently. DNs combine characteristics of both undirected and directed graphical models. The dependencies among variables are represented with an undirected graph, so conditional independence can be interpreted using graph separation. However, as with directed models, dependencies are quantified using conditional probability distributions (CPDs) of a variable given its parents. The primary distinction between DNs and other graphical models is that DNs are an *approximate* model. Because the CPDs are learned independently, DN models are not guaranteed to specify a coherent probability distribution (see *Learning* section below for details).

DNs offer several advantages over conventional Bayesian networks, but also have several disadvantages (Heckerman et al. 2000). Unlike Bayesian networks, DNs are difficult to construct using a knowledge-based approach and they cannot represent causal relationships. However, DN models can encode predictive relationships (i.e. dependence and independence) and there are simple techniques for parameter estimation and structure learning of DNs.

The characteristics that distinguish DN models from Bayesian networks make them similar to Markov random fields. Both DNs and Markov random fields use undirected graphs to represent dependencies among variables. When the causal relationships among variables are unknown, undirected models are often more interpretable than directed models which require d-separation reasoning to assess conditional independencies. Undirected graphical models also have straightforward techniques for parameter estimation and structure learning (e.g. Della Pietra, Della Pietra and Lafferty 1997). DN conditional probability distributions will generally be easier to inspect and understand than Markov network clique potentials, but DNs approximate the full joint distribution and therefore Markov networks may produce more accurate probabilities.

Representation. The DN model consists of a set of conditional probability distributions (CPDs), one for each variable given its parents. Consider the set of variables $X=(X_1,\ldots,X_n)$ with a joint distribution $p(x)=p(x_1,\ldots,x_n)$. A dependency network for X is represented by a graph G where each node in G corresponds to an X_i□X. The parents of node X_i, denoted pa_i, consist of the nodes in its *Markov blanket*: This specifies that, given its parents, a node is conditionally independent of the rest of the nodes in the graph:

$$p(x_i \mid pa_i) = p(x_i \mid \mathbf{x} \setminus x_i)$$

The undirected edges of the graph connect each node x_i to each of its parents (the nodes in pa_i). Furthermore, each node in the graph contains a local CPD, $p_i=p(x_i|pa_i)$. Together these CPDs specify the joint distribution over $\boldsymbol{X}$.

For example, the DN in figure 1 could be used to model the set of variables $\boldsymbol{X}=(X_1,X_2,X_3,X_4,X_5)$. Each node is conditionally independent of the other nodes in the graph given its parents. For example, X_1 is conditionally independent of X_2 and X_4 given X_3 and X_5. Each node contains a CPD, which specifies a probability distribution over possible values given the values of its parents. The full joint probability is the product of the local CPDs:

$$p(X) = p(X_1 \mid X_3,X_5)p(X_2 \mid X_3,X_4)p(X_3 \mid X_1,X_2)p(X_4 \mid X_2,X_3)p(X_5 \mid X_1)$$

Notice that the model may have cycles. For example, X_2, X_3 and X_4 form a clique in the graph. This necessitates the use of approximate inference techniques (see *Inference* section below for details). Also, notice that X_4 is dependent on X_3, but the reverse is not true. An undirected edge is placed between two nodes if either of the nodes is dependent on the other. In this case, a node's parents will still form a Markov blanket, but they won't be the minimal set (e.g. $p_3=p(x_3|x_1,x_2,x_4)= p(x_3|x_1 ,x_2)$).

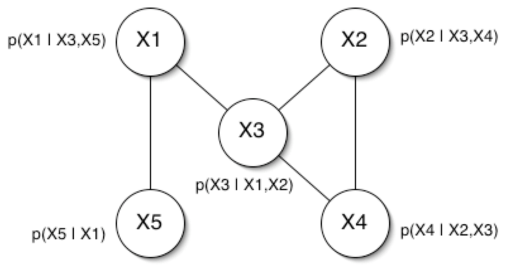

Fig. 1. Sample dependency network.

Learning. As with any graphical model, there are two components to learning a DN: structure learning and parameter estimation. Both structure learning and parameter estimation is accomplished through learning the local CPDs of each variable. The structure of a DN model is defined by the components of the local CPDs, much as feature specifications define the structure of a undirected graphical model. The edges of the graph connect a node to each of the variables in its local CPD. The parameters of the model correspond to the parameters of the local CPDs.

The DN learning algorithm learns a separate CPD for each variable, conditioned on the other variables in the data. For the DN model to be less than fully connected, a selective learning algorithm should be used. Any selective modeling technique can be used, however, to build a parsimonious DN model it is desirable to use a selective learner that will learn small, accurate CPDs.

For example, an algorithm for learning relational probability trees could be used to model X_1 given X_2, X_3, X_4, X_5. The variables included in the tree would then be designated as X_1's parents in the network, the appropriate edges would be added to the graph (e.g. X_1 to X_3 and X_1 to X_5) and the tree itself would be used as the local CPD for X_1. Learning the full DN in figure 1 would require learning five models, one tree for each variable in the graph.

Although the DN approach to structure learning is simple, it can result in an inconsistent network, both structurally and numerically—there may be no joint distribution from which each of the CPDs can be obtained using the rules of probability. Learning each local CPD independently may result in an inconsistent network where there is an edge between two variables but one is not dependent on the other (e.g. X_4 is dependent on X_3 but not vice versa). Independent parameter estimation may also result in inconsistencies where the overall joint distribution does not sum to 1. However, Heckerman et al. (2000) show that DNs will be "nearly" consistent if learned from large data sets. That is, the data serve a coordinating function that ensures some degree of consistency among the CPDs.

Inference. Unlike Bayesian networks, the dependency network graphs are potentially cyclic. This is due to the nature of the structure-learning algorithm where there is no restriction on the form of the CPDs. Cyclic dependencies necessitate the use of approximate inference techniques such as Gibbs sampling (e.g. Neal 1993) or loopy belief propagation (e.g. Murphy, Weiss and Jordan 1999). Heckerman et al. use Gibbs sampling to combine the models to approximate the full joint distribution and extract probabilities of interest. In practice, DNs have produced good approximations to the joint distributions represented by directed graphical models (Heckerman et al. 2000).

3.2 Relational Dependency Networks

Relational dependency networks (RDNs) extend DNs to work with relational data. The extension is similar to the way in which probabilistic relational models (Getoor et al. 2001) extend Bayesian networks for relational data.

Representation. RDNs specify a probability model over a network of instances. Given a set of objects and the links between them, a RDN defines a full joint probability distribution over the attribute values of the objects. Attributes of an object can depend probabilistically on other attributes of the object, as well as on attributes of objects in its relational neighborhood.

Instead of defining the dependency structure over attributes, as in DNs, RDNs define a generic dependency structure at the level of object *types*. Each attribute A_i associated with object type X is linked to a set of parents that influence the value of

A_i. Parents of A_i are either (1) other attributes associated with type X, or (2) attributes associated with objects of type Y where objects Y are linked to objects X. For the latter type of dependency, if the relation between X and Y is one-to-many, the “parent” consists of a set of attribute values. In this situation, RDNs uses aggregated features to map sets of values into single values.

For example, Figure 2 contains an example dataset from the movie domain. There are three types of objects—movies, studios and actors. Each object type has a number of associated attributes that will be each be modeled in the RDN. Consider the attribute *actor.hasAward*—the set of potential parents for this attribute consists of *actor.gender*, *movie.receipts* and *studio.country*. Notice that an actor can star in multiple movies and thus be associated indirectly with multiple studios.

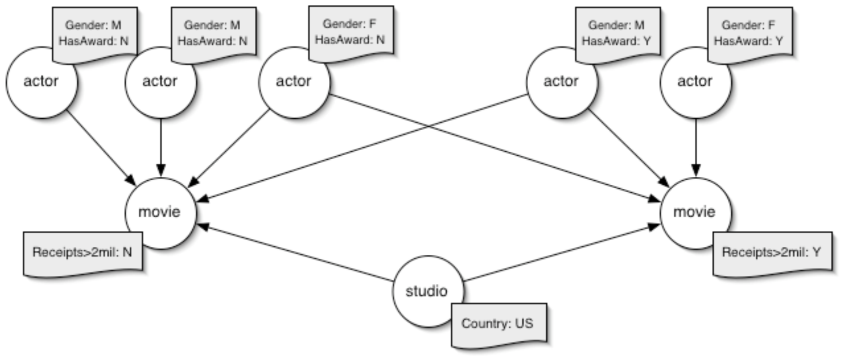

Fig. 2. Sample relational data graph.

The full RDN model 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 the data graph. Consequently, the total number of nodes in the model graph will be $\sum_{\mathrm{T}} N_{\mathrm{T}}A_{\mathrm{T}}$ where N_{T} is the number of objects of type T in the data graph and A_{T} is the number of attributes for objects of that type. To make the models tractable, the structure and parameters of the CPDs are tied—the RDN model contains a single CPD template for each attribute of each type of object. For the example above, the model would consist of four CPDs, one for *actor.gender*, *actor.hasAward*, *movie.receipts* and *studio.country*.

To construct the model graph, the set of template CPDs is *rolled-out* over the entire data graph. Each object-attribute pair gets a separate, local copy of the appropriate CPD. This 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 (e.g. hidden Markov models, PRMs).

Figure 3 contains a possible RDN model graph for the example discussed above. It shows dependencies between *actor.gender* and *actor.hasAward*, between *actor.hasAward* and *movie.receipts*, and between *movie.receipts* and *studio.country*. Furthermore, there is a dependency between *movie.receipts* of related movies. Notice the hyper-edges between movies and associated actor awards. This indicates that movie receipts is dependent on an aggregated feature of *actor.hasAward* (e.g. EXISTS(*actor.hasAward*=Y)). Aggregation is one approach to ensure the template CPDs are applicable across data graphs of varying structure. Each movie may have a different number of actor award values, so aggregation is used to map these sets of values into single values.

Learning. Learning an RDN model again consists of two tasks: learning the dependency structure, and estimating the parameters of the conditional probability distributions. The RDN learning algorithm is much like the DN learning algorithm, except we use a selective *relational* classification algorithm to learn a set of conditional models. We use relational probability trees (RPTs) for the CPD components of the RDN. RPTs extend standard probability estimation trees to a relational setting in which data instances are heterogeneous and interdependent (Neville et al. 2003). RPT models estimate probability distributions over possible class label values in the same manner as conventional classification trees, but the algorithm looks beyond the attributes of the item for which the class label is defined and considers the effect of the local relational neighborhood on the probability distribution. RPT models represent a series of questions to ask about an item and the objects/links in its relational neighborhood.

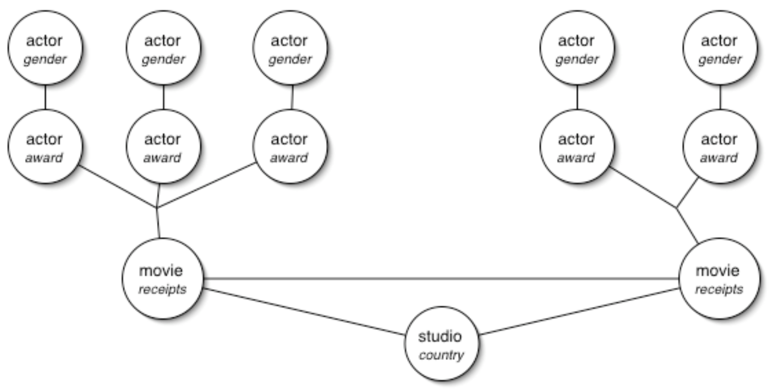

Fig. 3. Sample RDN model graph.

The RPT learning algorithm (Neville et al. 2003) uses a novel form of randomization tests to adjust for biases towards particular features due to the characteristics of the relational data (e.g. degree disparity, autocorrelation). We have shown that RPTs learned with randomization tests, build significantly smaller trees than other models and achieve equivalent, or better, performance. This characteristic of the RPTs is crucial for learning parsimonious RDN models—the collection of RPT models will be used during Gibbs sampling so the size of the models will have a direct impact on inference efficiency.

Given a data graph, an object type and a target attribute—an RPT is learned to predict the attribute given (1) the other attributes of the object, and (2) the attributes of other objects and links in the relational neighborhood. In our current approach, the user specifies the size of relational neighborhood to be considered by the RPT algorithm in (2). For efficiency reasons, we've limited the algorithm to consider attributes of objects one or two links away in the data graph. However, it is possible for the RPT models to consider attributes of much more "distant" objects (in the sense of a graph neighborhood).

Figure 4 shows an example RPT learned on data from the IMDb to predict whether a movie's opening-weekend receipts are more than $2million, given the attributes of movies and everything up to two links away—actors, directors, producers and studios, as well as movies associated with those objects (see Section 4 for experimental details). The tree indicates that movie *receipts* depend on the receipts of other movies made by the same studio, as well as actor age and movie genre. The root node of this tree asks whether more than 60% of the other movies made by the studio (e.g. *Studio Movie* objects) have *receipts=Y*. If this is not the case, the model moves to the next node on the right branch and asks whether the movie has more than 5 actors born after 1958. If the answer to this question is also no, a prediction of *P(Y)*=0.01 is returned.

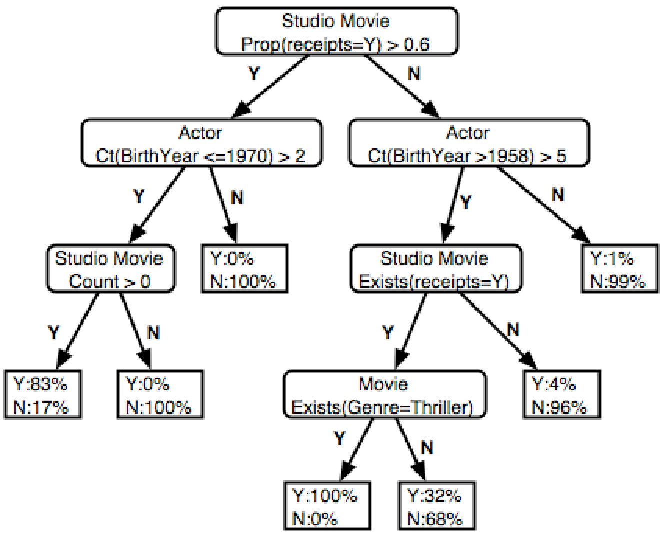

Fig. 4. Example RPT learned for the IMDb dataset to predict movie *receipts* given the labels of related movies.

Inference. As in the case of DNs, we use Gibbs sampling for inference in RDN models. For classification tasks, we want to estimate the full joint distribution over the target attributes in the graph. During inference, the model graph consists of a rolled-out network with both observed and unobserved variables. For example, we may want to use the network in Figure 3 to predict *movie.receipts* given *studio.country*, *actor.gender* and *actor.hasAward*. In this case, all attribute values are observed except *movie.receipts*. The values of the target variable are initialized to values drawn from the prior distribution (e.g. default distribution of *movie.receipts* in the training set). Gibbs sampling then proceeds iteratively, estimating the full joint distribution in cycles. For each target variable, the RPT model is used to return a probability distribution given the current attribute values in the rest of the graph. A new value is drawn from the distribution, assigned to the variable and recorded. This process is repeated for each unobserved variable in the graph. After a sufficient number of iterations, the values will be drawn from a stationary distribution and we can use the samples to estimate the full joint distribution. There are many implementation issues that could improve the estimates obtained from a Gibbs sampling chain, such as length of "burn in" and number of samples. We have not yet investigated the effects of these decisions on RDN performance. For the experiments reported in this paper we do not use a burn-in period and we use a fixed length chain of 2000 samples.

4 Experiments

This paper focuses on evaluation of RDNs in a classification context, where only a single attribute is unobserved in the test set—others are assumed to be observed and are not modeled with CPDs.

The experiments reported below are intended to evaluate two assertions. The first claim is that dependencies among instances can be used to improve model accuracy. We evaluate this claim by comparing the performance of two models. The first model is a conventional RPT model—an individual classification model that does not use labels of related instances, reasoning about each instance independently from other instances. We call this model *RPT-indiv*. The second model is a collective classification RDN model that exploits additional information available in labels of related instances and reasons about networks of instances collectively.

The second claim is that the RDN models, using Gibbs sampling, can effectively infer labels for a network of instances. To evaluate this claim, we include two more models for comparison. The third model is a conventional RPT model in which we allow the true labels of related instances to be used during both learning and inference. We call this model *RPT-ceiling*. This model is included as a ceiling comparison for the RDN model. It shows the level of performance possible if the model knew the *true* labels of related instances. The fourth model is intended to assess the need for a collective inference procedure. We call this model *RPT-default*. It reports the performance achieved on the first round of Gibbs sampling. This is equivalent to learning a conventional RPT model with the true labels of related instances, then randomly assigning labels according to the prior distribution of labels to use for inference. Since the test sets have connections to labeled instances in the training set (see next section for details), it is possible that these labels could provide enough information for accurate inferences, making Gibbs sampling unnecessary.

4.1 Tasks

The first data set is drawn from the Internet Movie Database (IMDb) (www.imdb.com). We gathered a sample of all movies released in the United States from 1996 through 2000, with opening weekend receipt information. The resulting collection contains 1383 movies. In addition to movies, the data set contains associated actors, directors, producers, and studios. In total, the data set contains 46,000 objects and 68,000 links. The learning task was to predict movie opening-weekend box office receipts. We discretized the attribute so that a positive label indicates a movie that garnered more than $2 million in opening-weekend box office receipts (*receipts*) (*P(Y)=0.45*).

We created training set/test set splits by temporal sampling into five sets, one for each year. Links to the future were removed from each sample. For example, the sample for 1997 contains links to movies from 1996, but not vice versa. We trained on the movies from one year (e.g. 1996) and tested on movies from the following year (e.g. 1997). Notice that during inference, there are links from the test set to fully labeled instances in the training set. This approach to sampling is intended to reproduce the expected domain of application for these models.

The RPT learning algorithm was applied to subgraphs centered on movies. The subgraphs consisted of movies and everything up to two links away—actors, directors, producers and studios, as well as movies associated with those objects. Nine attributes were supplied to the models, including studio country, actor birth-year and the class label of related movies two links away. Figure 4 shows an example RPT learned with the class labels of related movies. For a given movie *x*, the objects tagged "Studio Movie" refer to the other movies made by the primary studio associated with *x*.

The second data set is drawn from Cora, a database of computer science research papers extracted automatically from the web using machine learning techniques (McCallum, Nigam, Rennie and Seymore 1999). We selected the set of 1478 machine-learning papers published from 1994 through 1998, along with associated authors, journals, books, publishers, institutions and references. The resulting collection contains 11,500 objects and 26,000 links. The prediction task was to identify paper topic. Machine learning papers are divided into seven topics {Reinforcement Learning, Case-Based Reasoning, Probabilistic Methods, Theory, Genetic Algorithms, Neural Networks, and Rule Learning}.

As with the IMDb, we created training set/test set splits by temporal sampling into five sets, one for each year. The RPT learning algorithm used subgraphs centered on papers. The subgraphs consisted of papers, authors, journals, books, publishers, institutions and references, as well as papers associated with the authors. Twelve attributes were available to the models, including the journal affiliation, paper venue, and the topic of papers one link away (references) and two links away (through authors). Figure 5 shows an example RPT learned with the topics of related papers.

The RPT learning algorithm used randomization tests for feature selection and a Bonferroni-adjusted p-value growth cutoff of □=0.05. The RDN algorithm used a fixed number of Gibbs sampling iterations (2000).

4.2 Results and Discussion

Table 1 shows accuracy results for each of the four models on the IMDb classification tasks. On the IMDb, the RDNs models perform comparably to the *RPT-ceiling* models. This indicates that the RDN model realized its full potential, reaching the same level of performance as if it had access to the *true* labels of related movies.

In addition, the performance of the RDN models is superior to both the *RPT-indiv* models (RPT learned without labels) and the *RPT-default* models (RPT learned with labels and tested with a default labeling for the class values of related instances). In the example RPT in Figure 4 we can see that two of the five features refer to the target label on related movies. This indicates that autocorrelation is both present in the data and identified by the RPT models. The performance improvement over *RPT-indiv* is due to successful exploitation of this autocorrelation.

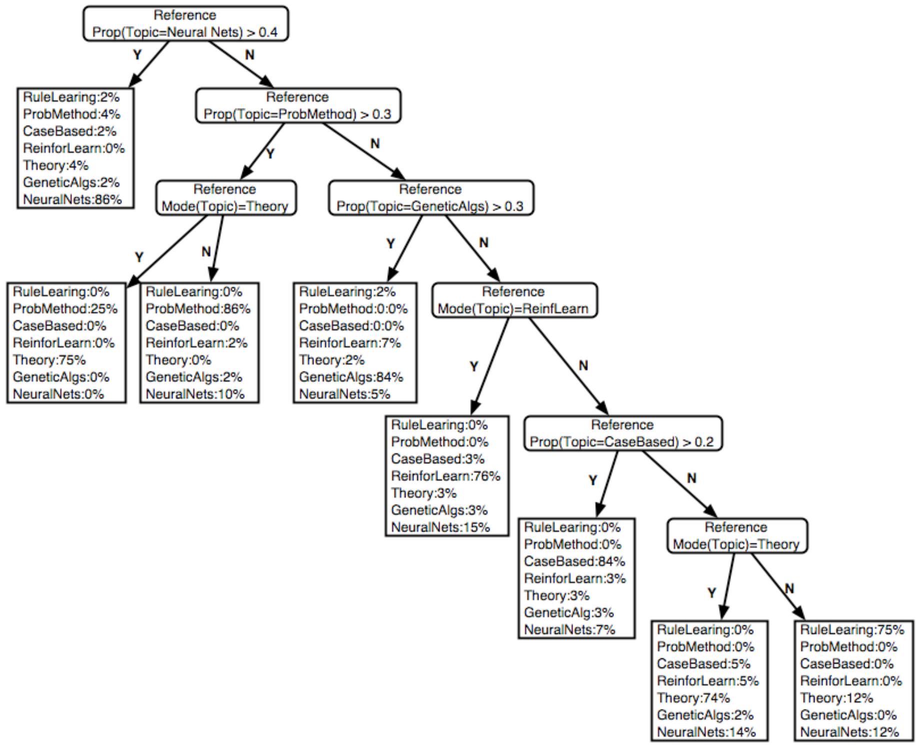

Fig. 5. Example RPT learned for the Cora dataset to predict paper topic given the topics of related papers.

We used two-tailed, paired t-tests to assess the significance of the accuracy results obtained from the four trials. The t-tests compare the RDN results to each of the three other models. The null hypothesis is that there is no difference in the accuracies of the two models; the alternative is that there is a difference. The resulting p-values are reported in the bottom row of the table, in the column of the model being compared to the RDN. The results support our conclusions above that the RDN results are significantly better than both *RPT-indiv* and *RPT-default*. Although the average performance of the RDNs is slightly lower than the *RPT-ceiling* models, the t-test indicates that this difference is not significant.

Table 1: Accuracy results for IMDb task

	RPT-indiv	*RPT-ceiling*	*RPT-default*	*RDN*
1	0.7148	0.8303	0.7473	0.8628
2	0.7500	0.8052	0.7370	0.8084
3	0.6893	0.8357	0.7429	0.7857
4	0.7103	0.8318	0.7383	0.8224
Avg	0.7161	0.8258	0.7414	0.8198
t-Test	0.0113	0.7529	0.0137	

On the Cora classification task, shown in Table 2, the RDN models show significant gains over the *RPT-indiv* and *RPT-default* models. This indicates that most of the predictive information lies in the topics of related pages. Nearly 90% of the features selected for the trees involved the topic of referenced papers. (Recall that reference topic was one of twelve attributes available to the model to form features.)

In this experiment, RDN models did not achieve the level of performance possible if the true label of related papers were known. However, the improvement from *RPT-indiv* and *RPT-default* models is notable. This is due to the paucity of predictive attributes other than the target label, clearly showing how autocorrelation can be exploited to improve model accuracy.

Table 2: Accuracy results for Cora task

	RPT-indiv	*RPT-ceiling*	*RPT-default*	*RDN*
1	0.2813	0.7437	0.4429	0.6852
2	0.2456	0.7646	0.4429	0.7013
3	0.2619	0.8027	0.5374	0.7313
4	0.2689	0.8571	0.5630	0.7983
Avg	0.2644	0.7920	0.4966	0.7290
t-Test	0.0004	0.0002	0.0004	

The difference in accuracy between the RDN and *RPT-ceiling* models may indicate that Gibbs sampling had not converged within the 2000 trials. To investigate this possibility we tracked accuracy throughout the Gibbs sampling procedure. Figure 6 shows curves for each of the tasks based on number of Gibbs iterations. The four lines represent each of the trials. Although we used 2000 iterations, we limit the graph because the accuracy plateaus within the first 150 iterations. Accuracy improves very quickly, leveling within the first 50 iterations. This shows that the approximate inference techniques employed by the RDN may be quite efficient to use in practice. We are currently experimenting with longer Gibbs chains, random restarts and convergence metrics to fully assess this aspect of the model.

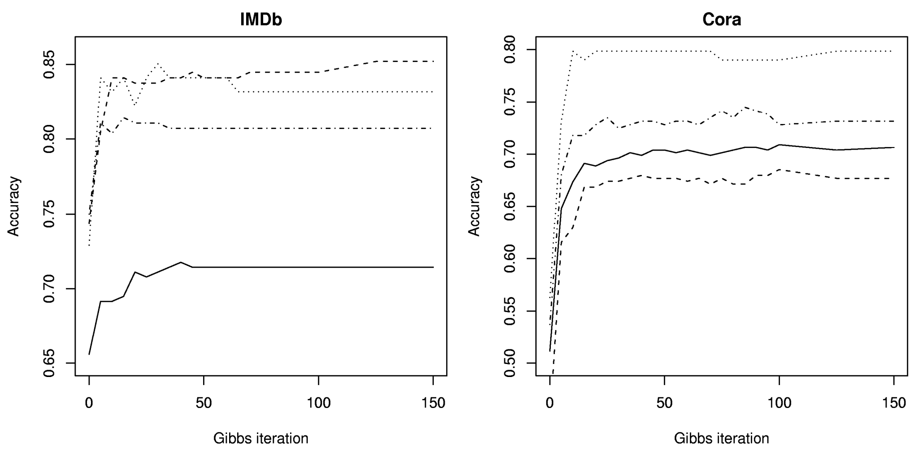

Fig. 6. Accuracy vs. number of Gibbs iterations. Each curve represents a separate trial.

If we can conclude from the learning curves that the Gibbs chain had converged, why didn't the RDN model perform as well as the *RPT-ceiling* model on Cora? One possible explanation is the lack of predictive attributes other than *topic*. The Gibbs chain may mix slowly, making the procedure unlikely to jump to distant portion of the labeling space. The inference procedure will suffer when there are no predictive attributes known with certainty to drive the mixing process in the right direction.

5 Conclusions and Future Work

These results indicate that collective classification with RDNs can offer significant improvement over non-collective approaches to classification when autocorrelation is present in the data. The performance of RDNs can approach the performance that would be possible if all the class labels of related instances were known. In addition, RDNs offer a relatively simple method for learning the structure and parameters of a graphical model, and they allow us to exploit existing techniques for learning conditional probability distributions. Here we have chosen to exploit our prior work on RPTs, which construct particularly parsimonious models of relational data, but we expect that the general properties of RDNs would be retained if other approaches to learning conditional probability distributions were used, given that those approaches are both selective and accurate.

Acknowledgments

Andrew McCallum provided valuable comments and suggestions on this work. We thank Michael Hay and Pippin Wolfe for comments. This research is supported under a National Science Foundation Graduate Research Fellowship and by DARPA, NSF and AFRL under contract numbers F30602-00-2-0597, EIA9983215 and F30602-01-2-0566. 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 the DARPA, the AFRL or the U.S. Government.

References

Bernstein, A., S. Clearwater, and F. Provost (2003). The relational vector-space model and industry classification. CDeR working paper #IS-03-02, Stern School of Business, New York University.

Chakrabarti, S., B. Dom and P. Indyk (1998). Enhanced hypertext categorization using hyperlinks. *Proc of ACM SIGMOD98*, pp 307-318.

Cortes, C., D. Pregibon, and C. Volinsky (2001). Communities of interest. *Proceedings Intelligent Data Analysis 2001.*

Della Pietra, S. V. Della Pietra and J. Lafferty (1997). Inducing features of random fields. I*EEE Transactions on Pattern Analysis and Machine Intelligence*, 19(4): 380-393.

Domingos, P., M. Richardson. Mining the Network Value of Customers (2001). *Proceedings of the 7th International Conference on Knowledge Discovery and Data Mining*, pp. 57-66.

Getoor, L., N. Friedman, D. Koller, and A. Pfeffer (2001). Learning probabilistic relational models. In *Relational Data Mining*, Dzeroski and Lavrac, Eds., Springer-Verlag.

Heckerman, D., D. Chickering, C. Meek, R. Rounthwaite, and C. Kadie (2000). Dependency networks for inference, collaborative filtering and data visualization. *JMLR*, 1:49--75.

Jensen, D. and J. Neville (2002). Linkage and autocorrelation cause bias in relational feature selection. *Machine Learning: Proceedings of the Nineteenth International Conference*. Morgan Kaufmann.

Jensen, D., J. Neville and M. Hay (2003). Avoiding bias when aggregating relational data with degree disparity. *Proc. of the 20th Intl Joint Conf. on Machine Learning*, to appear.

Kleinberg, J. (1999). Authoritative sources in a hyper-linked environment. *Journal of the ACM* 46:604-632.

Lafferty, J., A. McCallum and F. Pereira (2001). Conditional random fields: Probabilistic models for segmenting and labeling sequence data. *Proceedings of ICML2001.*

McCallum, A., K. Nigam, J. Rennie and K. Seymore (1999). A Machine Learning Approach to Building Domain-specific Search Engines. In *Proceedings of the 19th International Joint Conference on Artificial Intelligence*, pp. 662-667.

Murphy, K., Y. Weiss, and M. Jordan (1999). Loopy belief propagation for approximate inference: an empirical study. *Proceedings of the 15th Annual Conference on Uncertainty in Artificial Intelligence*.

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

Neville, J. and D. Jensen (2000). Iterative Classification in Relational Data. *AAAI Workshop on Learning Statistical Models from Relational Data*, 42-49.

Neville, J., D. Jensen, L. Friedland, M. Hay (2003). Learning relational probability trees. *Proceedings of the 9th International Conference on Knowledge Discovery & Data Mining*, to appear.

Taskar, B., P. Abbeel and D. Koller (2002). Discriminative probabilistic models for relational data. *Proceedings of UAI2002.*