Journal of Machine Learning Research vol (year) 00-00 Submitted 1/2005; Published 00/0000

Classification in Networked Data[0]: A toolkit and a univariate case study

Sofus A. Macskassy SMACSKAS@STERN.NYU.EDU
Foster Provost FPROVOST@STERN.NYU.EDU
New York University
44 W. 4th Street
New York, NY 10012

Editor:

Abstract

This paper presents NetKit, a modular toolkit for classification in networked data, and a case-study of its application to a collection of networked data sets used in prior machine learning research. Networked data are relational data where entities are interconnected, and this paper considers the common case where entities whose labels are to be estimated are linked to entities for which the label is known. NetKit is based on a three-component framework, comprising a local classifier, a relational classifier, and a collective inference procedure. Various existing relational learning algorithms can be instantiated with appropriate choices for these three components and new relational learning algorithms can be composed by new combinations of components. The case study demonstrates how the toolkit facilitates comparison of different learning methods (which so far has been lacking in machine learning research). It also shows how the modular framework allows analysis of subcomponents, to assess which, whether, and when particular components contribute to superior performance. The case study focuses on the simple but important special case of univariate network classification, for which the only information available is the structure of class linkage in the network (i.e., only links and some class labels are available). To our knowledge, no work previously has evaluated systematically the power of class-linkage alone for classification in machine learning benchmark data sets. The results demonstrate clearly that simple network-classification models perform remarkably well—well enough that they should be used regularly as baseline classifiers for studies of relational learning for networked data. The results also show that there are a small number of component combinations that excel, and that different components are preferable in different situations, for example when few versus many labels are known.

Keywords: relational learning, network learning, collective inference, collective classification, networked data

1. Introduction

This paper is about classification of entities in *networked* data, one type of relational data. Relational classifier induction algorithms, and associated inference procedures, have been developed in a variety of different research fields and problem settings (Emde and Wettschereck, 1996; Flach and Lachiche, 1999; Dzeroski and Lavrac, 2001). Generally, these algorithms consider not only the features of the entities to be classified, but the relations to and the features of linked entities.

0. S.A. Macskassy and Provost, F.J., "Classification in Networked Data: A toolkit and a univariate case study" CeDER Working Paper CeDER-04-08, Stern School of Business, New York University, NY, NY 10012. December 2004.

Observed improvements in generalization performance demonstrate that taking advantage of relational information in addition to attribute-value information can improve performance—sometimes substantially (e.g. (Taskar et al., 2001; Jensen et al., 2004)).

Networked data are the special case of relational data where entities are interconnected, such as web-pages or research papers (connected through citations). This is in contrast with domains such as molecules or arches, where each entity is a self-contained graph and connections between the entities are absent or ignored. With a few exceptions (e.g., (Chakrabarti et al., 1998), (Taskar et al., 2001)), recent machine learning research on classification with networked data has focused on *across-network* inference: learning from one network and applying the learned models to a separate, presumably similar network (Craven et al., 1998; Lu and Getoor, 2003).

This paper focuses on *within-network* inference. In this case, networked data have the unique characteristic that training entities and entities whose labels are to be estimated are interconnected. Although the network may have disconnected components, generally there is not a clean separation between the entities for which class membership is known and the entities for which estimations of class membership are to be made. This introduces complications (Jensen and Neville, 2002b). For example, the usual careful separation of data into training and test sets is difficult. More important, thinking in terms of separating training and test sets obscures an important facet of the data: entities with known classifications can serve two roles. They act first as training data and subsequently as background knowledge during inference (Provost et al., 2003).

Many real-world problems, especially those involving social networks, exhibit opportunities for within-network classification. For example, in fraud detection entities to be classified as being fraudulent or legitimate are intertwined with those for which classifications are known. In counterterrorism and law enforcement, suspicious people may interact with known 'bad' people. Some networked data are by-products of social networks, rather than directly representing the networks themselves. For example, networks of web pages are built by people and organizations that are interconnected; when classifying web pages, some classifications (henceforth, *labels*) may be known and some may need to be estimated.

To our knowledge there has been no systematic study of machine learning methods for within-network classification that compares various algorithms on various data sets. A serious obstacle to undertaking such a study is the scarcity of available tools and source code, making it hard to compare various methodologies and algorithms. Such an in-depth study is further hindered by the fact that many relational learning algorithms can be separated into various sub-components. Ideally, a study should account for the contributions of the sub-components, and assess the relative advantage of alternatives. To enable such a study, we need a framework that facilitates isolating the performance of and interchanging sub-components.

As a main contribution of this paper, we introduce a network learning toolkit (NetKit-SRL) that enables in-depth, component-wise studies of techniques for statistical relational learning and inference with networked data. Starting with prior published work, we have abstracted the described algorithms and methodologies into a modular framework. The toolkit is based on this framework.[1]

NetKit is interesting for several reasons. First, it encompasses several currently available systems, which are realized by choosing particular instantiations for the different components. This allows us to compare and contrast the different systems on equal footing. Perhaps more importantly, the modularity of the toolkit broadens the design space of possible systems beyond those

1. NetKit-SRL, or NetKit for short, is written in Java 1.5 and is available as open source.

that have appeared in prior published work, either by mixing and matching the components of the prior systems, or by introducing new alternatives for components. Finally, NetKit's modularity not only allows allows for direct comparison of various models, but also for comparison of isolated components as we will show.

To illustrate, we use NetKit to conduct in an in-depth case study of within-network classification. The case study considers univariate learning and classification in homogeneous networks. We compare various techniques on twelve benchmark data sets from four domains used in prior machine learning research. Beyond illustrating the value of the toolkit, the case study makes several contributions. It provides systematic support for the claim that with networked data even univariate classification can be quite effective, and therefore it should be considered as a baseline against which to compare new relational learning algorithms (Macskassy and Provost, 2003). The case study illustrates a bias/variance tradeoff in networked classification, based on the principle of homophily (Blau, 1977; McPherson et al., 2001) (cf., assortativity (Newman, 2003) and autocorrelation (Jensen and Neville, 2002b). Indeed, the simplest method works so well it suggests that we should consider finding more diverse benchmark data sets. The case study also suggests network-classification analogues to feature selection and active learning.

The remainder of the paper is organized as follows. Section 2 describes the problem of network learning more formally, introduces the modular framework, reviews prior work, and describes NetKit. Section 3 covers the case study, including the experimental methodology, data used, toolkit components used, and the results and analysis of the comparative study. The paper ends with discussions of limitations and conclusions.

2. Network Learning

Traditionally, machine learning methods have treated entities as independent, which makes it possible to infer class membership on an entity-by-entity basis. With networked data, the class membership of one entity may have an influence on the class membership of a related entity. Furthermore, entities not directly linked may be related by chains of links, which suggests that it may be beneficial to infer the class memberships of all entities simultaneously. Collective inferencing in relational data (Taskar et al., 2002; Neville and Jensen, 2004) makes simultaneous statistical judgments regarding the values of an attribute or attributes for multiple entities in a graph G for which some attribute values are not known.

For the univariate case study presented below, the (single) attribute of vertex v_i, representing the class, can take on some categorical value $X \in \mathcal{X}$.

> Given graph $G = (V, E)$, a single attribute x_i for each vertex $v_i \in V$, and given known values for x_i for some subset of vertices V^K, *univariate collective inferencing* is the process of simultaneously inferring the values of x_i for the remaining vertices, $V^U = V - V^K$, or a probability distribution over those values.

As a shorthand, we will use $\mathbf{x}^K$ to denote the set (vector) of class values for V^K, and similarly for $\mathbf{x}^U$. Then, $G^K = (V, E, \mathbf{x}^K)$ denotes everything that is known about the graph (we do not consider the possibility of unknown edges). Edge $e_{ij} \in E$ represents the edge between vertices v_i and v_j, and w_{ij} represents the edge weight. For this paper we consider only undirected edges, simply ignoring directionality if necessary for a particular application.

Rather than estimating the full joint probability distribution $P(\mathbf{x}^U|G^K)$, relational learning often enhances tractability by making a Markov assumption:

$$P(x_i|G) = P(x_i|\mathcal{N}_i), \tag{1}$$

where $\mathcal{N}_i$ is the set of "neighbors" of vertex v_i such that $P(x_i|\mathcal{N}_i)$ is independent of $G - \mathcal{N}_i$ (i.e., $P(x_i|\mathcal{N}_i) = P(x_i|G)$). For this paper, we make the ("first-order") assumption that $\mathcal{N}_i$ comprises only the immediate neighbors of v_i in the graph. As one would expect, and as we will see, this assumption can be violated to a greater or lesser degree based on how edges are defined.

Given $\mathcal{N}_i$, a relational model can be used to estimate x_i. Note that $\mathcal{N}_i^U$ $(= \mathcal{N}_i \cap V^U)$—the set of neighbors of v_i whose values of attribute x are not known—could be non-empty. Therefore, even if the Markov assumption holds, a simple application of the relational model may be insufficient. However, the relational model may be used to estimate the labels of $\mathcal{N}_i^K = \mathcal{N}_i - N_i^U$. Further, just as estimates for the labels of $\mathcal{N}_i^U$ influence the estimate for x_i, x_i also influences the estimate of the labels of $\mathcal{N}_i^U$. In order to simultaneously estimate $\mathbf{x}^U$, various collective methods have been introduced for relational inference, including Gibbs sampling (Geman and Geman, 1984), loopy belief propagation (Pearl, 1988), relaxation labeling (Chakrabarti et al., 1998), and other iterative classification methods (Neville and Jensen, 2000; Lu and Getoor, 2003). All such methods require initial ("prior") estimates of the values for $P(\mathbf{x}^U|G^K)$. The priors could be Bayesian subjective priors (Savage, 1954), or they could be estimated from data. A common estimation method is to employ a non-relational learner, using available "local" attributes of v_i to estimate x_i (e.g., as done by Chakrabarti et al. (1998)). In the univariate case, such local attributes are not available; for our case study, we use the marginal class distribution over V^K as the prior for all $x_i \in \mathbf{x}^U$.

2.1 Network Learning Framework

As suggested by the discussion above, one prominent class of systems for learning and inference in networked data can be characterized by three main components. For each component, there are many possible instantiations.

1. **Non-relational ("local") model.** This component consists of a (learned) model, which uses only local information—namely information about (attributes of) the entities whose target variable is to be estimated. The local models can be used to generate priors that comprise the initial state for the relational learning and collective inference components. They also can be used as one source of evidence during collective inference. These models typically are produced by traditional machine learning methods.

2. **Relational model.** In contrast to the non-relational component, the relational model makes use of the relations in the network as well as the values of attributes of related entities, possibly through long chains of relations. Relational models also may use local attributes of the entities.

3. **Collective inferencing.** The collective inferencing component determines how the unknown values are estimated together, possibly influencing each other, as described above.

Certain techniques from prior work, described below, can be instantiated with particular choices of these components. For example, using a naive Bayes classifier as the local model, a naive Bayes

Markov Random Field classifier for the relational model, and relaxation labeling for the inferencing method forms the system used by Chakrabarti et al. (1998). Using logistic regression for the local and relational models, and iterative classification for the inferencing method produces Lu & Getoor's (2003) link-based classifier. Using class priors for the local model, a (weighted) majority vote of neighboring classes for the relational model, and relaxation labeling for the inference method forms Macskassy & Provost's (2003) relational neighbor classifier.

2.2 Prior Work

For machine learning research on networked data, the watershed paper of Chakrabarti et al. (1998) studied classifying web-pages based on the text and (possibly inferred) class labels of neighboring pages, using relaxation labeling paired with naive Bayes local and relational classifiers. In their experiments, using the link structure substantially improved classification over using the local (text) information alone. Further, considering the text of the neighbors generally hurt performance (based on the methods they used), whereas using only the (inferred) class labels improved performance.

More recently, Lu and Getoor (2003) investigated network classification applied to linked documents (web pages and published manuscripts with an accompanying citation graph). Similarly to the work of Chakrabarti et al. (1998), Lu and Getoor (2003) use the text of the document as well as a relational classifier. Their "link-based" classifier was a logistic regression model based on a vector of aggregations of properties of neighboring nodes linked with different types of links (in-, out-, co-links). Various aggregates were considered, such as the mode (the value of the most often occurring neighbor class), a binary vector with a value of 1 at cell i if there was a neighbor whose class label was c_i, and a count vector where cell i contained the number of neighbors belonging to class c_i. In their experiments, the count model performed best.

Univariate within-network classification has been considered previously (Bernstein et al., 2002, 2003; Macskassy and Provost, 2003). Using business news, Bernstein et al. (2003) linked companies if they co-occurred in a news story. They demonstrated the effectiveness of various vector-space techniques for network classification of companies into industry sectors, based on vectors of class labels of the neighbors. This work did not use collective inferencing, performing only a one-shot prediction based on the known neighborhood (knowing 90% of the class labels and predicting the remaining 10%). Other domains such as web-pages, movies and citation graphs have also been considered for univariate within-network classification; Macskassy and Provost (2003) investigated how well the univariate classification performed as varying amounts of data initially were labeled. They used a relaxation labeling method similar to that used by Chakrabarti et al. (1998). In both studies, a very simple model predicting class membership based on the most prevalent class in the neighborhood was shown to perform remarkably well. The present paper can be seen in part as a systematic followup to these workshop papers.

Markov Random Fields (MRFs) have been used extensively for univariate network classification for vision and image restoration. Nodes in the network are pixels in an image and the labels are image-related such as whether a pixel is part of a vertical or horizontal border (Dobrushin, 1968; Geman and Geman, 1984; Winkler, 2003). MRFs are used to estimate the joint probability of a set of nodes based on their immediate neighborhoods under the first-order Markov assumption that $P(x_i|\mathbf{X}/x_i) = P(x|\mathcal{N}_i)$, where $\mathbf{X}/x_i$ means all nodes in $\mathbf{X}$ except x_i and $\mathcal{N}_i$ is a neighborhood function returning the neighbors of v_i. Chakrabarti et al. (1998) use an MRF formulation for their network classifier (described above), which we reconstruct in NetKit.

One popular method to compute the MRF joint probability is Gibbs sampling (described below). The most common use of Gibbs sampling in vision is not to compute the final posteriors as we do in this paper, but rather to get final classifications. One way to enforce that the Gibbs sampler settles to a final state is by using a simulated annealing approach where the temperature is dropped slowly until nodes no longer change state (Geman and Geman, 1984). Neville and Jensen (2000) used a simulated annealing approach in their *iterative classification* collective inference procedure, where a label for a given node was kept only if the relational classifier was confident about the label at a given threshold, otherwise the label would be set to `null`. By slowly lowering this threshold, the system was eventually able to label all nodes. NetKit incorporates iterative classification based on the subsequent work of Lu and Getoor (2003) (described above).

Graph-cut techniques recently have been used in vision research as an alternative to using Gibbs sampling (Boykov et al., 2001), iteratively changing the labelings of many nodes at once by solving a min-cut/max-flow problem based on the current labelings. In addition to the explicit links in the data, each node is also connected to one special node per class label. A min-cut algorithm is then used to partition the graph such that only one class-node remains linked to each node in the data. Based on this cut, the method then changes the labelings, and repeats until no pixels change labels. These methods are very fast. NetKit does not yet incorporate graph-cut techniques.

Several recent methods apply to learning in networked data, beyond the homogeneous, univariate case treated in this paper. Conditional Random Fields (CRFs) (Lafferty et al., 2001) are an extension of MRFs where labels are conditioned not only on the labels of neighbors, but also on the attributes of the node itself and the attributes of the neighborhood nodes. CRFs were applied to part-of-speech (POS) tagging in text, where the nodes in the graphs represented the words in the sentence, connected by their word order. The labels to be predicted were POS-tags and the attribute of a node was the word it represents. The neighborhood of a word comprised the words on either side of it.

Relational Bayesian Networks (RBNs)[2] (Koller and Pfeffer, 1998; Friedman et al., 1999; Taskar et al., 2001) extend Bayesian networks (BNs (Pearl, 1988)) by taking advantage of the fact that a variable used in one instantiation of a BN may refer to the exact same variable in another BN. For example, if the grade of a student depends upon his professor, this professor is the same for all students in the class. Therefore, rather than building one BN and using it in isolation for each entity, RBNs directly link shared variables in "unrolled" BNs, thereby generating one big network of connected entities for which collective inferencing can be performed. Most relevant to this paper, for within-network classification RBNs were applied by Taskar et al. (2001) to various domains, including a data set of published manuscripts linked by authors and citations. Loopy Belief Propagation (Pearl, 1988) was used to perform the collective inferencing. The study showed that the PRM performed better than a non-relational naive Bayes classifier and that using both author and citation information in conjunction with the text of the paper worked better than using only author or citation information in conjunction with the text.

Relational Dependency Networks (RDNs) (Neville and Jensen, 2003, 2004), extend dependency networks (Heckerman et al., 2000) in much the same way that RBNs extend Bayes Networks. RDNs have been used successfully on a bibliometrics data set, a movie data set and a linked web-page data set, where they were shown to perform much better than a relational probability tree (RPT)

2. These originally were called Probabilistic Relational Models (PRMs). PRM now typically is used as a more general term which includes other models such as Relational Dependency Networks and Relational Markov Networks, described next.

Input: $G^K, V^U, \text{RC}_{\text{type}}, \text{LC}_{\text{type}}, \text{CI}_{\text{type}}$
Induce a local classification model, LC, of type LC_{type}, using G^K Induce a relational classification model, RC, of type RC_{type}, using G^K Estimate $x \in V^U$ using LC. Apply collective inferencing of type CI_{type}, using RC as the model.
Output: Final estimates for $x_i \in V^U$

Table 1: High-level pseudo code for the main core of the Network Learning Toolkit.

(Neville et al., 2003) using no collective inferencing. Gibbs sampling was used to perform collective inferencing.

Relational Markov Networks (RMNs) (Taskar et al., 2002) extend Markov Networks (Pearl, 1988). The clique potential functions used are based on functional templates, each of which is a (learned, class-conditional) probability function based on a user-specified set of relations. Taskar et al. (2002) applied RMNs to a set of web-pages and showed that they performed better than other non-relational learners as well as naive Bayes and logistic regression when used with the same relations as the RMN. Loopy Belief Propagation was used to perform collective inferencing.

The above systems use only a few of the many relational learning techniques proposed in the literature. There are many more, for example from the rich literature of inductive logic programming (ILP) (*e.g.* Flach and Lachiche (1999); Raedt et al. (2001); Dzeroski and Lavrac (2001); Kramer et al. (2001); Domingos and Richardson (2004)), or based on using relational database joins to generate relational features (e.g. Perlich and Provost (2003); Popescul and Ungar (2003); Perlich and Provost (2004)). These techniques could be the basis for additional relational model components in NetKit.

2.3 Network Learning Toolkit (NetKit-SRL)

NetKit is designed to accommodate the interchange of components and the introduction of new components. Any local model can be paired with any relational model, which can then be combined with any collective inference method. NetKit's core routine is simple and is outlined in Table 1.

NetKit consists of these primary modules:

1. **Input:** This module reads data into a memory-resident graph G.

2. **Local classifier inducer (LC):** Given as training data V^K, this module returns a model which using only attributes of a node $v_i \in V^U$ will estimate x_i. Ideally, LC will estimate a probability distribution over the possible values for x_i.

3. **Relational classifier inducer (RC):** Given G^K, this module returns a model which using v_i and $\mathcal{N}_i$ will estimate x_i. Ideally, RC will estimate a probability distribution over the possible values for x_i.

4. **Collective Inferencing:** Given a graph G possibly with some x_i known, a set of priors over $\mathbf{x}^U$, and a relational model $\mathcal{M}_R$, this applies collective inferencing to estimate $\mathbf{x}^U$.

5. **Weka Wrapper:** This module is a wrapper for Weka[3] (Witten and Frank, 2000) and will convert the graph representation of v_i into an entity that can either be learned from or be used to estimate x_i.

Implementation details on these modules can be found in Appendix B. The current version of NetKit-SRL, while able to read in heterogeneous graphs, only supports classification in graphs consisting of a single type of node.

2.4 NetKit Components

This section describes the particular relational classifiers and collective inference methods implemented in NetKit for the univariate case study. First, we describe the four (univariate[4]) relational classifiers (RC components). Then, we describe the three collective inference methods.

2.4.1 RELATIONAL CLASSIFIERS (RC)

All four relational classifiers take advantage of a first-order Markov assumption on the network: only a node's local neighborhood is necessary for classification. The univariate case renders this assumption particularly restrictive: only the class labels of the local neighbors are necessary. The local network is defined by the user, analogous to the user's definition of the feature set for propositional learning. Entities whose class labels are not known are either ignored or are assigned a prior, depending upon the choice of local classifier.

2.4.1.1 WEIGHTED-VOTE RELATIONAL NEIGHBOR CLASSIFIER (WVRN)

Our first and simplest classifier (cf., Macskassy and Provost (2003)[5]) estimates class-membership probabilities based on one assumption in addition to the Markov assumption: the entities exhibit homophily—i.e., linked entities have a propensity to belong to the same class (Blau, 1977; McPherson et al., 2001). This homophily-based model is motivated by observations and theories of social networks (Blau, 1977; McPherson et al., 2001), where homophily is ubiquitous. Homophily was one of the first characteristics noted by early social network researchers (Almack, 1922; Bott, 1928; Richardson, 1940; Loomis, 1946; Lazarsfeld and Merton, 1954), and holds for a wide variety of different relationships (McPherson et al., 2001). It seems reasonable to conjecture that homophily may also be present in other sorts of networks, especially networks of artifacts created by people. (Recently *assortativity*, a link-centric notion of homophily, has become the focus of mathematical studies of network structure (Newman, 2003).)

Definition. Given $v_i \in V^U$, the weighted-vote relational-neighbor classifier (wvRN) estimates $P(x_i|\mathcal{N}_i)$ as the (weighted) mean of the class-membership probabilities of the entities in $\mathcal{N}_i$:

$$P(x_i = X|\mathcal{N}_i) = \frac{1}{Z} \sum_{v_j \in \mathcal{N}_i} w_{i,j} \cdot P(x_j = X|\mathcal{N}_j), \tag{2}$$

where Z is the usual normalizer. As the above is a recursive definition (for undirected graphs, $v_j \in \mathcal{N}_i \Leftrightarrow v_i \in \mathcal{N}_j$) the classifier uses the "current" estimate for $P(x_j = X|\mathcal{N}_j)$, where the "current" estimate is defined by the collective inference technique being used.

3. We use version 3.4.2. Weka is available at `http://www.cs.waikato.ac.nz/~ml/weka/`
4. The open-source NetKit release contains multivariate versions of these classifiers.
5. Previously called the probabilistic Relational Neighbor classifier (pRN).

2.4.1.2 CLASS-DISTRIBUTION RELATIONAL NEIGHBOR CLASSIFIER (CDRN)

Learning a model of the distribution of neighbor class labels may lead to better discrimination than simply using the (weighted) mode. Following Perlich and Provost (2003), and in the spirit of Rocchio's method (Rocchio, 1971), we define node v_i's class vector $\mathrm{CV}(v_i)$ to be the vector of summed linkage weights to the various (known) classes, and class X's reference vector $\mathrm{RV}(X)$ to be the average of the class vectors for nodes known to be of class X. Specifically:

$$\mathrm{CV}(v_i)_k = \sum_{v_j \in \mathcal{N}_i, x_j = X_k} w_{i,j}, \tag{3}$$

where $\mathrm{CV}(v_i)_k$ represents the k^{th} position in the class vector and X_k is the k^{th} class. Based on these class vectors, the reference vectors can then be defined as the vector sum:

$$\mathrm{RV}(X) = \frac{1}{|V_X^K|} \sum_{v_i \in V_X^K} \mathrm{CV}(v_i), \tag{4}$$

where $V_X^K = \{v_i | v_i \in V^K, x_i = X\}$.

During training, neighbors in V^U are ignored. For prediction, estimated class membership probabilities are used for neighbors in V^U, and equation (3) becomes:

$$\mathrm{CV}(v_i)_k = \sum_{v_j \in \mathcal{N}_i} P(x_j = X_k | \mathcal{N}_j) \cdot w_{i,j} \tag{5}$$

Definition. Given $v_i \in V^U$, the class-distribution relational-neighbor classifier (cdRN) estimates the probability of class membership, $P(x_i = X | \mathcal{N}_i)$, by the normalized vector distance between v_i's class vector and class X's reference vector:

$$P(x_i = X | \mathcal{N}_i) = \frac{1}{Z} \mathrm{dist}(\mathrm{CV}(v_i), \mathrm{RV}(X)), \tag{6}$$

where Z is the usual normalizer and $\mathrm{dist}(a, b)$ is any vector distance function (L_1, L_2, cosine, etc.). For the results presented below, we use cosine distance.

As with wvRN, Equation 5 is a recursive definition, and therefore the value of $P(x_j = X | \mathcal{N}_j)$ is approximated by the "current" estimate as given by the selected collective inference technique.

2.4.1.3 NETWORK-ONLY BAYES CLASSIFIER (NBC)

NetKit's network-only Bayes classifier (nBC) is based on the algorithm described by Chakrabarti et al. (1998). To start, assume there is a single node v_i in V^U. The nBC uses multinomial naive Bayesian classification based on the classes of v_i's neighbors.

$$P(x_i = X | \mathcal{N}_i) = \frac{P(\mathcal{N}_i | X) \cdot P(X)}{P(\mathcal{N}_i)}, \tag{7}$$

where

$$P(\mathcal{N}_i | X) = \frac{1}{Z} \prod_{v_j \in \mathcal{N}_i} P(x_j = X_{j^*} | x_i = X)^{w_{i,j}} \tag{8}$$

where Z is a normalizing constant and X_{j^*} is the class observed at node v_j. Because $P(\mathcal{N}_i)$ is the same for each class, normalization across the classes allows us to ignore it (as with naive Bayes generally).

We call nBC "network-only" to emphasize that in the application to the univariate case study below, we do not use local attributes of a node. As discussed above, Chakrabarti et al. initialize nodes' priors based on a naive Bayes model over the local document text.[6] In the univariate setting, local text is not available. We therefore use the same scheme as for the other RCs: initialize unknown labels as decided by the local classifier being used (either the class prior or 'null', depending on the CI component, as described below). If a neighbor's label is 'null', then it is ignored for classification. Also, Chakrabarti et al. differentiated between incoming and outgoing links, whereas we do not. Finally, Chakrabarti et al. do not mention how or whether they account for possible zeros in the estimations of the marginal conditional probabilities; we apply traditional Laplace smoothing where $m = |\mathcal{X}|$, the number of classes.

The foregoing assumes all neighbor labels are known. When the values of some neighbors are unknown, but estimations are available, we follow Chakrabarti et al. (1998) and perform Markov Random Fields (MRF) estimations (Dobrushin, 1968; Geman and Geman, 1984; Winkler, 2003), based on how different configurations of neighbors' classes affect a target entity's class. Specifically, the classifier computes a Bayesian combination based on (estimated) configuration priors and the entity's known neighbors. Chakrabarti et al. (1998) describe this procedure in detail. For our case study, such an estimation is necessary only when using relaxation labeling (described below).

2.4.1.4 NETWORK-ONLY LINK-BASED CLASSIFICATION (NLB)

The final relational classifier used in the case study is a network-only derivative of the link-based classifier (Lu and Getoor, 2003). The network-only Link-Based classifier (nLB) creates a feature vector for a node by aggregating the labels of neighboring nodes, and then uses logistic regression to build a discriminative model based on these feature vectors. This learned model is then applied to estimate $P(x_i = X|\mathcal{N}_i)$. As with the nBC, the difference between the "network-only" link-based classifier and Lu and Getoor's version is that for the univariate case study we do not consider local attributes (e.g., text).

As described above, Lu and Getoor (2003) considered various aggregation methods: existence (binary), the mode, and value counts. The last aggregation method, the count model, is equivalent to the class vector $\mathrm{CV}(v_i)$ defined in Equation 5. This was the best performing method in the study by Lu and Getoor, and is the method on which we base nLB. The logistic regression classifier used by nLB is the multiclass implementation from Weka version 3.4.2.

We made one minor modification to the original link-based classifier. Perlich (2003) argues that in different situations it may be preferable to use vectors based on raw counts (as given above) or vectors based on normalized counts. We did preliminary runs using both. The normalized vectors generally performed better, and so we use them for the case study.

2.4.2 COLLECTIVE INFERENCE METHODS (CI)

This section describes three collective inferencing (CI) methods implemented in NetKit and used in the case study. As described above, given (i) a network initialized by the local model, and (ii) a relational model, a CI method infers a set of class labels for $\mathbf{x}^U$, ideally with the maximum joint probability. Alternatively, if estimates of entities' class-membership probabilities are needed, the

6. The original classifier was defined as: $P(x_i = X|\mathcal{N}_i) = P(\mathcal{N}_i|X) \cdot P(\tau_i|v_i) \cdot P(X)$, with τ_i being the text of the document-entity represented by vertex v_i.

1. Initialize priors using the local classifier model, $\mathcal{M}_L$. For $v_i \in V^U$, $\mathbf{c}_i \leftarrow \mathcal{M}_L(v_i)$, where $\mathbf{c}_i$ represents the estimate of $P(x_i)$. For the case study, the local classifier model returns the marginal class distribution estimated from $\mathbf{x}^K$.
2. Generate a random ordering, O, of vertices in V^U.
3. Set initial labels in O by sampling based on the priors. This will generate a particular configuration of labels in G.
4. For elements $v_i \in O$ in order:
   (a) Apply the relational classifier model: $\mathbf{c}_i \leftarrow \mathcal{M}_R$.
   (b) Sample a value x_s from $\mathbf{c}_i$.
   (c) Set $x_i \leftarrow x_s$.
   Note that when $\mathcal{M}_R$ is applied to x_i it uses the "new" labelings from elements $1, \ldots, (i-1)$, while using the "current" labelings for elements $(i+1), \ldots, n$.
5. Repeat prior step 200 times without keeping any statistics. This is known as the burnin period.
6. Repeat again for 2000 iterations, counting the number of times each x_i is assigned a particular value $X \in \mathcal{X}$. Normalizing these counts forms the final class probability estimates.

Table 2: Pseudo-code for Gibbs sampling.

CI method estimates the marginal probability distribution $P(x_i|G^K, \Lambda)$ for each $x_i \in \mathbf{x}^U$, where Λ stands for the priors returned by the local classifier.

2.4.2.1 GIBBS SAMPLING (GS)

Gibbs sampling (GS) (Geman and Geman, 1984) is commonly used for collective inferencing with relational learning systems. The algorithm is straightforward and is shown in Table 2.[7] The use of 200 and 2000 for the burnin period and number of iterations are commonly used values.[8] Ideally, we would iterate until the estimations converge. Although there are convergence tests for the Gibbs sampler, they are not robust nor well understood (cf. Gilks et al. (1995)), so we simply use a fixed number of iterations.

Notably, because all nodes are assigned a class at every iteration, when GS is used the relational models will always see a fully labeled/classified neighborhood, making prediction straightforward. For example, nBC does not need its MRF estimation.

2.4.2.2 RELAXATION LABELING (RL)

The second collective inferencing method implemented and used in this study is relaxation labeling (RL), based on the method of Chakrabarti et al. (1998). Rather than treat G as being in a specific labeling "state" at every point (as Gibbs sampling does), relaxation labeling retains the uncertainty, keeping track of the current probability estimations for $\mathbf{x}^U$. The relational model must

7. This instance of Gibbs sampling uses a single random ordering ("chain"), as this is what we used in the case study. In the case study, using 10 chains (the default in NetKit) had no effect on the final accuracies.
8. As it turns out, in our case study GS invariably reached a seemingly final plateau in fewer than 1000 iterations, and often in fewer than 500.

1. For $v_i \in V^U$, initialize the prior: $\mathbf{c}_i^{(0)} \leftarrow \mathcal{M}_L(v_i)$. For the case study, the local classifier model returns the class priors.
2. For elements $v_i \in V^U$:
   (a) Estimate x_i by applying the relational model:
   $$\mathbf{c}_i^{(t+1)} \leftarrow \mathcal{M}_R(v_i^{(t)}), \tag{9}$$
   where $\mathcal{M}_R(v_i^{(t)})$ denotes using the estimates $\mathbf{c}^{(t)}$, and t is the iteration count. This has the effect that all predictions are done pseudo-simultaneously based on the state of the graph after iteration t.
3. Repeat for T iterations, where $T = 99$ for the case study. $\mathbf{c}^{(T)}$ will comprise the final class probability estimations.

Table 3: Pseudo-code for Relaxation Labeling.

be able to use these estimations. Further, rather than estimating one node at a time and updating the graph right away, relaxation labeling "freezes" the current estimations so that at step $t + 1$, all vertices will be updated based on the estimations from step t. The algorithm is shown in Table 3.

Preliminary runs showed that RL sometimes does not converge, but rather ends up oscillating between two points.[9] NetKit performs simulated annealing—on each subsequent iteration giving more weight to a node's own current estimate and less to the influence of its neighbors.

The new updating step, replacing Equation 9, is defined as:

$$\mathbf{c}_i^{(t+1)} = \beta^{(t+1)} \cdot \mathcal{M}_R(v_i^{(t)}) + (1-\beta^{(t+1)}) \cdot \mathbf{c}_i^{(t)}, \tag{10}$$

where

$$\begin{aligned} \beta^0 &= k \\ \beta^{(t+1)} &= \beta^{(t)} \cdot \alpha, \end{aligned} \tag{11}$$

where k is a constant, which for the case study we set to 1.0, and α is a decay constant, which we set to 0.99. Preliminary testing showed that final performance is very robust as long as $0.9 < \alpha < 1$. Smaller values of α can lead to neighbors losing their weight too quickly, which can hurt performance when only very few labels are known. A post-mortem of the results showed that the accuracies often converged within the first 20 iterations.

2.4.2.3 ITERATIVE CLASSIFICATION (IC)

The third and final collective inferencing method implemented in NetKit and used in the case study is the variant of Iterative Classification described in the work on link-based classification (Lu and Getoor, 2003) and shown in Table 4. As with Gibbs sampling, the relational model never sees uncertainty in the labels of (neighbor) entities. Either the label of a neighbor is `null` and ignored (which only happens in the first iteration), or it is assigned a definite label.

9. Such oscillation has been noted elsewhere for closely related methods (Murphy et al., 1999).

1. For $v_i \in V^U$, initialize the prior: $\mathbf{c}_i \leftarrow \mathcal{M}_L(v_i)$. The link-based classification work of Lu and Getoor (2003) uses a local classifier to set initial classifications. This will clearly not work in our case (all unknowns would be classified as the majority class), and we therefore use a local classifier model which returns `null` (*i.e.*, it does not return an estimation.)
2. Generate a random ordering, O, of elements in V^U.
3. For elements $v_i \in O$:
   (a) Apply the relational classifier model, $\mathbf{c}_i \leftarrow \mathcal{M}_R$, using all non-null labels. Entities which have not yet been classified will be ignored (this will only occur in the first iteration).
   (b) Classify v_i:
   $$x_i = \text{argmax}_X \mathbf{c}_i.$$
4. Repeat for $T = 1000$ iterations, or until no entities receive a new class label.[a] The estimates from the final iteration will be used as the final class probability estimates.

a. A post-mortem of the results showed that IC often stopped within $10 - 20$ iterations when paired with cdRN, nBC or nLB. For wvRN, it generally ran the full 1000 iterations, although the accuracy quickly plateaued and wvRN ended up moving within a small range of similar accuracies.

Table 4: Pseudo-code for Iterative Classification.

3. Case Study

The study presented in this section has two goals. First, it showcases NetKit, demonstrating that the modular framework indeed facilitates the comparison of systems for learning and inference in networked data. Second, it examines the simple-but-important special case of univariate learning and inference in homogeneous networks, comparing alternative techniques that have not before been compared systematically, if at all. The setting for the case study is simple: For some entities in the network, the value of x_i is known; for others it must be estimated.

Univariate classification, albeit a simplification for many domains, is important for several reasons. First, it is a representation that is used in some applications. In the introduction we mentioned fraud detection. As a specific example, a telephone account that calls the same numbers as a known fraudulent account (and hence the accounts are connected through these intermediary numbers) is suspicious (Fawcett and Provost, 1997; Cortes et al., 2001). For phone fraud, univariate network classification often provides alarms with reasonable coverage and remarkably low false-positive rates. In fact, the fraud detection work of Cortes et al. focuses on exactly this representation (albeit also considering changes in the network over time). Generally speaking, a homogeneous, univariate network is an inexpensive (in terms of data gathering, processing, storage) approximation of many complex networked data problems. Fraud detection applications certainly do have a variety of additional attributes of importance; nevertheless, univariate simplifications are very useful and are used in practice.

The univariate case also is important scientifically. It isolates a primary difference between networked data and non-networked data, facilitating the analysis and comparison of relevant classification and learning methods. One thesis of this study is that there is considerable information

Category	Size
High-revenue	572
Low-revenue	597
Total	1169
Base accuracy	51.07%

Table 5: Details on class distribution for the IMDb data set.

inherent in the structure of the networked data and that this information can be readily taken advantage of, using simple models, to estimate the labels of unknown entities. This thesis is tested by isolating this characteristic—namely ignoring any auxiliary attributes and only allowing the use of known class labels—and empirically evaluating how well univariate models perform in this setting on benchmark data sets.

Considering homogeneous networks plays a similar role. Although the domains we consider have obvious representations consisting of multiple entity types and edges (e.g., people and papers for node types and same-author-as and cited-by as edge types in a citation-graph domain), a homogeneous representation is much simpler. In order to assess whether a more complex representation is worthwhile, it is necessary to assess standard techniques on the simpler representation (as we do in this case study). Of course, the way a network is "homogenized" may have a considerable effect on classification performance. We will revisit this below in Section 3.3.6.

3.1 Data

The case study reported in this paper makes use of 12 benchmark data sets from four domains that have been the subject of prior study in machine learning. As this study focuses on networked data, any singleton (disconnected) entities in the data were removed. Therefore, the statistics we present may differ from those reported previously.

3.1.1 IMDB

Networked data from the Internet Movie Database (IMDb)[10] have been used to build models predicting movie success based on box-office receipts (Jensen and Neville, 2002a). Following the work of Neville et al. (2003), we focus on movies released in the United States between 1996 and 2001 with the goal of estimating whether the opening weekend box-office receipts "will" exceed $2 million (Neville et al., 2003). Obtaining data from the IMDb web-site, we identified 1169 movies released between 1996 and 2001 that we were able to link up with a high-revenue classification in the database given to us by the authors of the original study. The class distribution of the data set is shown in Table 5.

We link movies if they share a production company, based on observations from previous work[11] (Macskassy and Provost, 2003). The weight of an edge in the resulting graph is the number of production companies two movies have in common. Notably, we ignore the temporal aspect of the movies in this study, simply labeling movies at random for the training set. This can lead to a movie in the test set being released earlier than a movie in the training set.

10. `http://www.imdb.com`
11. And on a suggestion from David Jensen.

Category	Size
Case Based	402
Genetic Algorithms	551
Neural Networks	1064
Probabilistic Methods	529
Reinforcement Learning	335
Rule Learning	230
Theory	472
Total	3583
Base accuracy	29.70%

Table 6: Details on class distribution for the CoRA data set.

3.1.2 CORA

The CoRA data set (McCallum et al., 2000) comprises computer science research papers. It includes the full citation graph as well as labels for the topic of each paper (and potentially sub- and sub-sub-topics).[12] Following a prior study (Taskar et al., 2001), we focused on papers within the machine learning topic with the classification task of predicting a paper's sub-topic (of which there are seven). The class distribution of the data set is shown in Table 6.

Papers can be linked in one of two ways: they share a common author, or one cites the other. Following prior work (Lu and Getoor, 2003), we link two papers if one cites the other. This number ordinarily would only be zero or one unless the two papers cite each other.

3.1.3 WEBKB

The third domain we draw from is based on the WebKB Project (Craven et al., 1998).[13] It consists of sets of web pages from four computer science departments, with each page manually labeled into 7 categories: course, department, faculty, project, staff, student or other. As with other work (Neville et al., 2003; Lu and Getoor, 2003), we ignore pages in the "other" category except as described below.

From the WebKB data we produce eight networked data sets for within-network classification. For each of the four universities, we consider two different classification problems: the 6 class problem, and following a prior study, the binary classification task of predicting whether a page belongs to a student (Neville et al., 2003).[14] The binary task results in an approximately balanced class distribution.

Following prior work on web-page classification, we link two pages by co-citations (if x links to z and y links to z, then x and y are co-citing z) (Chakrabarti et al., 1998; Lu and Getoor, 2003). To weight the link between x and y, we sum the number of hyperlinks from x to z and separately the number from y to z, and multiply these two quantities. For example, if student x has 2 edges to a group page, and a fellow student y has 3 edges to the same group page, then the weight along that path between those 2 students would be 6. This weight represents the number of possible co-citation paths between the pages. Co-citation relations are not uniquely useful to domains involving documents; for example, as mentioned above, for phone-fraud detection bandits often call the same

12. These labels were assigned by a naive Bayes classifier (McCallum et al., 2000).
13. We use the WebKB-ILP-98 data.
14. It turns out that the relative performance of the methods is quite different on these two variants.

Class	Number of web-pages			
	Cornell	**Texas**	**Washington**	**Wisconsin**
student	145	163	151	155
not-student	201	171	283	193
Total	346	334	434	348
Base accuracy	58.1%	51.2%	60.8%	55.5%

Table 7: Details on class distribution for the WebKB data set using binary class labels.

Category	Number of web-pages			
	cornell	**texas**	**washington**	**wisconsin**
course	54	51	170	83
department	25	36	20	37
faculty	62	50	44	37
project	54	28	39	25
staff	6	6	10	11
student	145	163	151	155
Total	346	334	434	348
Base accuracy	41.9%	48.8%	39.2%	44.5%

Table 8: Details on class distribution for the WebKB data set using 6-class labels.

numbers as previously identified bandits. We chose co-citations for this case study based on the prior observation that a student is more likely to have a hyperlink to her advisor or a group/project page rather than to one of her peers (Craven et al., 1998).[15]

To produce the final data sets, we extracted the pages that have at least one incoming and one outgoing link. We removed pages in the "other" category from the classification task, although they were used as "background" knowledge—allowing 2 pages to be linked by a path through an "other" page. For the binary tasks, the remaining pages were categorized into either student or not-student. The composition of the data sets is shown in Tables 7 and 8.

3.1.4 INDUSTRY CLASSIFICATION

The final domain we draw from involves classifying public companies by industry sector. Companies are linked via cooccurrence in text documents. We create two different data sets, representing different sources and distributions of documents and different time periods (which correspond to different topic distributions).

INDUSTRY CLASSIFICATION (YH)

As part of a study of activity monitoring (Fawcett and Provost, 1999), Fawcett and Provost collected $22,170$ business news stories from the web between 4/1/1999 and 8/4/1999. Following the study by Bernstein et al. (2003) discussed above, we identified the companies mentioned in each story and added an edge between two companies if they appeared together. The weight of an edge is the number of such cooccurrences found in the complete corpus. The resulting network comprises

15. We return to these data in Section 3.3.5, where we show and discuss how using the hyperlinks directly is not sufficient for any of the univariate methods to do well.

Sector	Number of companies
Basic Materials	104
Capital Goods	83
Conglomerates	14
Consumer Cyclical	99
Consumer NonCyclical	60
Energy	71
Financial	170
Healthcare	180
Services	444
Technology	505
Transportation	38
Utilities	30
Total	1798
Base accuracy	28.1%

Table 9: Details on class distribution for industry-yh data set.

Sector	Number of companies
Basic Materials	83
Capital Goods	78
Conglomerates	13
Consumer Cyclical	94
Consumer NonCyclical	59
Energy	112
Financial	268
Healthcare	279
Services	478
Technology	609
Transportation	47
Utilities	69
Total	2189
Base accuracy	27.8%

Table 10: Details on class distribution for the industry-pr data set.

1798 companies which cooccurred with at least one other company. To classify a company, we used Yahoo!'s 12 industry sectors. Table 9 shows the details of the class memberships.

INDUSTRY CLASSIFICATION (PR)

The second Industry Classification data set is based on 35,318 prnewswire press releases gathered from April 1, 2003 through September 30, 2003. As above, the companies mentioned in each press release were extracted and an edge was placed between two companies if they appeared together in a press release. The weight of an edge is the number of such cooccurrences found in the complete corpus. The resulting network comprises 2189 companies which cooccurred with at least one other company. To classify a company, we use the same classification scheme from Yahoo! as before. Table 10 shows the details of the class memberships.

3.2 Experimental Methodology

NetKit allows for any combination of a local classifier (LC), a relational classifier (RC) and a collective inferencing method (CI). If we consider an LC-RC-CI configuration to be a complete network-classification (NC) method, we have 12 to compare on each data set. Since, for this paper, the LC component is directly tied to the CI component, we henceforth consider an NC system to be an RC-CI configuration.

We first verify that the network structure alone (linkages plus known class labels) often contains a considerable amount of useful information for entity classification. To that end, we assess the classification performance of each NC as we vary from 10% to 90% the percentage of nodes in the network for which class membership is known initially. Varying the amount of information initially available assesses: (1) whether the network structure enables classification; (2) how much prior information is needed in order to perform well, and (3) whether there are regular patterns of improvement with more labeled entities.

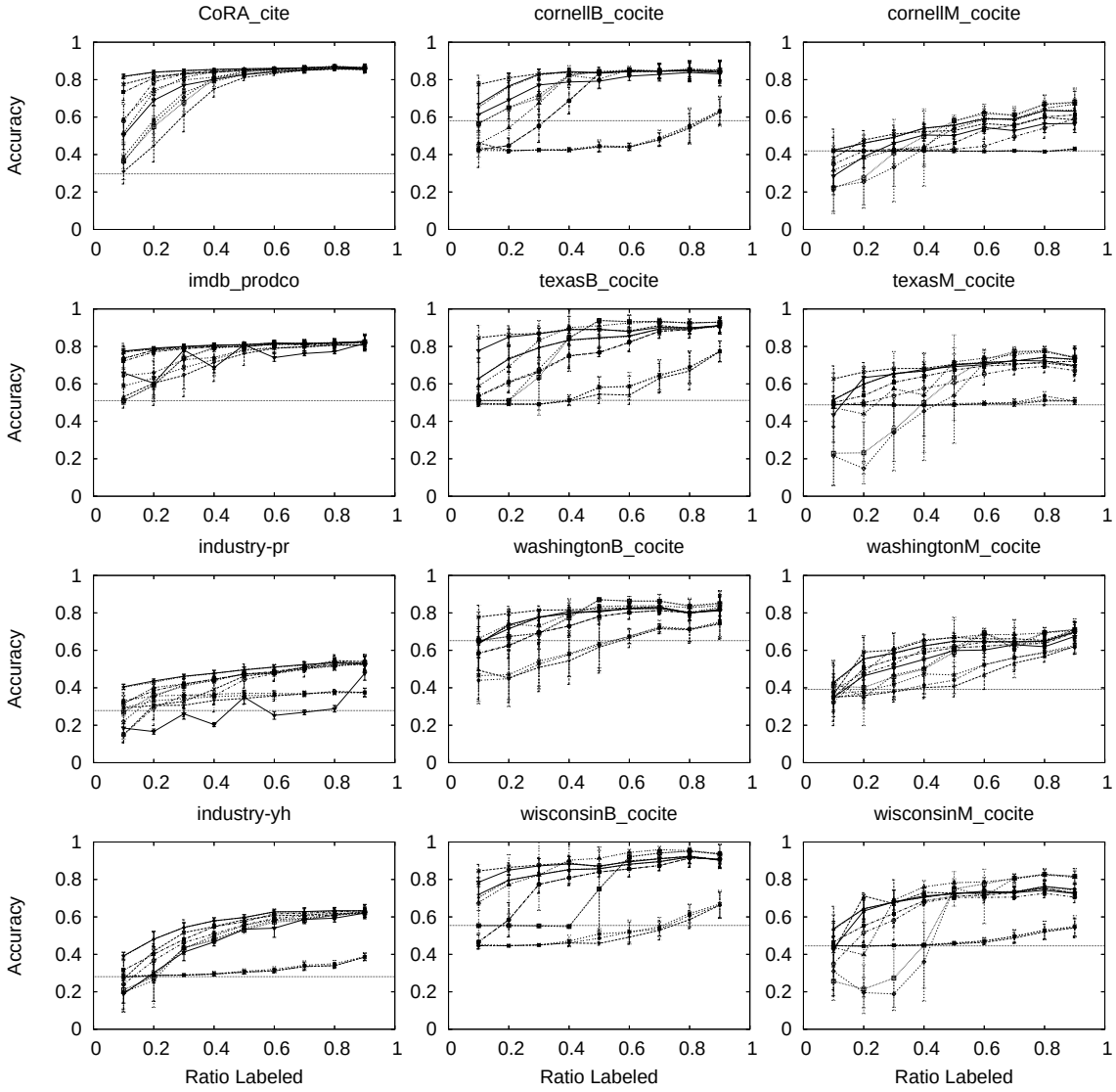

Figure 1: Overall classification accuracies on the twelve data sets. Horizontal lines represent predicting the most prevalent class. Individual methods will be clarified in subsequent figures. The horizontal axis plots the fraction (r) of a network's nodes for which the class label is known ex ante. In each case, when many labels are known (right end) there is a set of methods that performs well. When few labels are known (left end) there is much more variation in performance. Data sets are tagged based on the edge-type used, where 'prodco' is short for 'production company', and 'B' and 'M' in the WebKB data sets represents 'binary' and 'multi-class' classifications, respectively.

Accuracy is averaged over 10 runs. Specifically, given a data set, $G = (V, E)$, the subset of entities with known labels V^K (the "training" data set[16]) is created by selecting a class-stratified random sample of $(100 \times r)\%$ of the entities in V. The test set, V^U is then defined as $V - V^K$. We further prune V^U by removing all nodes in *zero-knowledge* components—nodes for which there is no path to any node in V^K. We use the same 10 training/test partitions for all NC systems.

3.3 Results

3.3.1 INFORMATION IN THE NETWORK STRUCTURE

Figure 1 shows the accuracies of the 12 NC systems across the 12 data sets as the fraction (r) of entities for which class memberships are known increases from $r = 0.1$ to $r = 0.9$. As mentioned above, in the univariate case, if the linkage structure is not known the only non-subjective alternative is to estimate using the class base rate (prior), which is shown by the horizontal line in the graphs. As is clear from Figure 1, many of the data sets contain considerable information in the class-linkage structure. The worst relative performance is on industry-pr, where at the right end of the curves the error rate nonetheless is reduced by 30–40%. The best performance is on webkb-texas, where the best methods reduce the error rate by close to 90%. And in most cases, the better methods reduce the error rate by over 50% toward the right end of the curves.

Machine learning studies on networked data sets seldom compare to simple network-classification methods like these, opting instead for comparing to non-relational classification. These results argue strongly that comparisons also should be made to univariate network classification, if the purpose is to demonstrate the power of a more sophisticated relational learning method.

3.3.2 COLLECTIVE INFERENCE COMPONENT

We now compare the different collective inference components. We are not aware of theory that makes a strong case for when one method should perform better than another. However, we will be comparing classification accuracy (rather than the quality of the probability estimates), so one might expect iterative classification to outperform Gibbs sampling and relaxation labeling, since the former focuses explicitly on maximum a posteriori (MAP) classification and the latter two focus on estimating the joint probability distribution over the class labels. On the other hand, with few known labels, MAP classifications may be highly uncertain, and it may be better to propagate uncertainty, as does relaxation labeling.

Figure 2 shows, for three of the data sets, the comparative performances of the three collective inference (CI) components. Each graph is for a particular relational classifier. The graphs show that, while the three CI components often perform similarly, their performances are clearly separated for low values of r.

Table 11 shows the p-values for a paired t-test assessing whether the first method (listed in column 1) is significantly better than the second. Specifically, for a given data set and label ratio (r), each NC experiment consisted of 10 random train/test splits—the same for all configurations. For each pair of CI components, pooling the 10 splits across the 4 RC components and 12 data sets yields 480 paired data points. The results show clearly that RL, across the board, outperformed both

16. These data set will be used not only for training models, but also as existing background knowledge during classification.

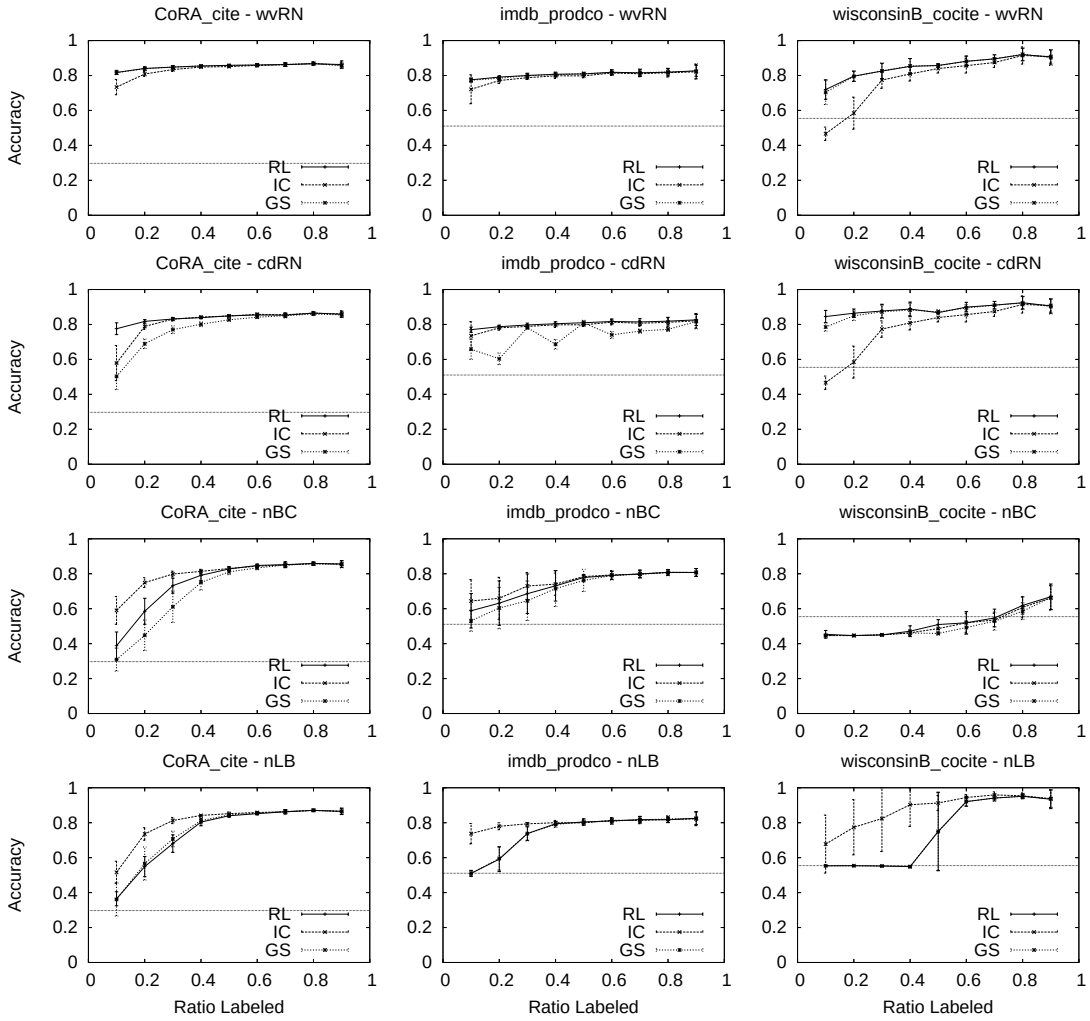

Figure 2: Comparison of Collective Inference methods on a select few data sets, with data set and RC component listed above each graph. The horizontal line represents predicting the most prevalent class.

GS and IC, often at $p \leq 0.001$. Further, we see that IC also was often better than GS, although not always significantly.

The foregoing shows that relaxation labeling is consistently better when the results are pooled across CI pairs. Table 12 shows the magnitude of the differences. In order to be comparable across data sets with different base rates, the table shows how much of an error reduction over the base rate the first method (listed first in column 1) produces as compared to the second (listed second in column 1). As a simple example, assume the base error rate is 0.4, method A yields an error rate of 0.1, and method B yields an error rate of 0.2. Method A reduces the error by 75%. Method B reduces the error by 50%. The relative error reduction of A vs. B is 1.5 (50% more error reduction). More precisely, for each labeling ratio r, we computed the relative error reduction ratio, $\mathrm{ER}^{*}_{\mathrm{REL}}$,

	sample ratio								
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
RL v GS	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
RL v IC	**0.001**	**0.001**	**0.100**	**0.025**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
IC v GS	*0.200*	**0.300**	**0.100**	**0.050**	*0.450*	**0.200**	**0.100**	**0.005**	**0.250**

Table 11: p-values for the statistical significance in differences in performance between pairs of CI components across all data sets and RC methods. For each cell, bold text means that the first method was better than the second method and italics means it was worse.

	sample ratio									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	**overall**
RL v GS	**2.790**	**1.462**	**1.136**	**1.124**	**1.063**	**1.061**	**1.042**	**1.035**	**1.014**	**1.093**
RL v IC	**404.315**	**1.593**	**1.115**	**1.078**	**1.072**	**1.055**	**1.037**	**1.018**	**1.013**	**1.098**
IC v GS	*144.937*	*1.090*	**1.019**	**1.043**	*1.009*	**1.005**	**1.005**	**1.016**	**1.002**	*1.004*

Table 12: Relative error reduction (ER^*_{REL}) improvements for each CI component across all data sets. Each cell shows the ratio of the better method's error reduction over the other method's error reduction. As above, bold text means that the first method was better than the second, and italics mean it was worse. The last column, overall, is based on taking the ratio of the average error reduction for the methods across all sample ratios.

between two components, CI_A and CI_B as follows.

$$\text{ER}_{\text{ABS}}(\text{RC,CI}, D, r) = (\text{base\_err}(D) - \text{err}(\text{RC-CI}, D, r)) \quad (12)$$

$$\text{ER}_{\text{REL}}(\text{RC,CI}, D, r) = \begin{cases} \text{NA} & \text{if } \text{ER}_{\text{ABS}}(\text{RC,CI}, D, r) < 0 \\ \frac{\text{ER}_{\text{ABS}}(\text{RC,CI},D,r)}{\text{base\_err}(D)} & \text{otherwise} \end{cases} \quad (13)$$

$$\text{ER}_{\text{REL}}(\text{RC,CI}, r) = \frac{1}{|\mathbf{D}|} \sum_{D \in \mathbf{D}} \text{ER}_{\text{REL}}(\text{RC,CI}, D, r) \quad (14)$$

$$\text{ER}_{\text{REL}}(\text{CI}, r) = \frac{1}{|\mathbf{RC}|} \sum_{\text{RC} \in \mathbf{RC}} \text{ER}_{\text{REL}}(\text{RC,CI}, r) \quad (15)$$

$$\text{ER}^*_{\text{REL}}(\text{CI}_A, \text{CI}_B, r) = \begin{cases} \infty & \text{if } \text{ER}_{\text{REL}}(\text{CI}_B, r) = \text{NA or } 0 \\ \frac{\text{ER}_{\text{REL}}(\text{CI}_A,r)}{\text{ER}_{\text{REL}}(\text{CI}_B,r)} & \text{otherwise} \end{cases} \quad (16)$$

$$(17)$$

where err(RC-CI,D, r) is the error for the configuration (RC and CI) on data set D with $r\%$ of the graph being labelled. A ratio $\rho > 1$ means that CI_A reduced the error by $(100 \times (1 - \rho))\%$ over that of CI_B.

Table 12, following the same layout as Table 11, shows the ratios for each CI comparison. The unusually large entries occur when $\text{ER}_{\text{REL}}(\text{CI}_B, r)$ is very close to zero. As is clear from this table, RL outperformed IC across the board, from as low as a 1.3% improvement ($r = 0.90$) to as high as 59% or better improvement ($r \leq 0.2$) when averaged over all the data sets and RC methods. Overall

	sample ratio									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	**total**
RL	11	11	10	7	4	5	6	4	6	64
GS	1	1	0	1	5	3	4	4	4	23
IC	0	0	2	4	3	4	2	4	2	21

Table 13: Number of times each CI method was the best across the 12 data sets.

RL improved performance over IC by about 10% as seen in the last column in the "RL v IC" row of the table. RL's advantage over IC improves monotonically as less is known in the network. Similar numbers and a similar pattern are seen for RL versus GS. IC and GS are comparable.[17]

The results so far have compared the CI methods disregarding the RC component. Table 13 shows, for each ratio as well as a total across all ratios, the number of times each CI implementation took part in the best-performing NC combination for each of the twelve data sets. Specifically, for each sampling ratio, each win for an RC-CI configuration counted as a win for the CI module of the pair (as well as a win for the RC module in the next section). For example, in Figure 2, the first column of four graphs shows the performances of the 12 NC combinations on the CoRA data; at the left end of the curves, wvRN-RL is the best performing combination. Table 13 adds further support to the conclusion that relaxation labeling (RL) was the overall best component, primarily due to its advantage at low values of r. We also see again that Gibbs Sampling (GS) and Iterative Classification (IC) were comparable.

3.3.3 RELATIONAL MODEL COMPONENT

Comparing relational models, we would expect to see a certain pattern: if even moderate homophily is present in the data, we would expect wvRN to perform well. Its nonexistent training variance[18] should allow it to perform relatively well, even with small numbers of known labels in the network. The higher-variance nLB may perform relatively poorly with small numbers of known labels (primarily because of the lack of training data, rather than problems with collective inference). On the other hand, wvRN is potentially a very-high-bias classifier (it does not learn at all). The learning-based classifiers may well perform better with large numbers of known labels if there are patterns beyond homophily to be learned. As a worst case for wvRN, consider a bipartite graph between two classes. In a leave-one-out cross-validation, wvRN would be wrong on every prediction. The relational learners should notice the true pattern immediately.

Figure 3 shows for four of the data sets the performances of the four RC implementations. The rows of graphs correspond to data sets and the columns to the three different collective inference methods. The graphs show several things, which will be clarified next. As would be expected, accuracy improves as more of the network is labeled, although in certain cases classification is remarkably accurate with very few known labels (e.g., see CoRA). One method is substantially worse than the others. Among the remaining methods, performance often differs greatly with few known labels, and tends to converge with many known labels. More subtly, there often is a crossing of curves when about half the nodes are labeled (e.g., see Washington).

17. NB: it is possible for the winner in Table 11 and the winner in Table 12 to disagree (as seen for the IC and GS comparisons at $r = 0.20$), because the relative error reduction depends on the base error whereas the statistical test is based on the absolute values.
18. NB: there still will be variance due to the set of known labels.

	sample ratio								
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
wvRN v cdRN	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
wvRN v nBC	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
wvRN v nLB	**0.001**	**0.001**	**0.001**	**0.001**	*0.300*	*0.001*	*0.001*	*0.001*	*0.001*
cdRN v nBC	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**
cdRN v nLB	**0.001**	**0.001**	**0.001**	**0.001**	*0.002*	*0.001*	*0.001*	*0.001*	*0.001*
nLB v nBC	*0.250*	**0.010**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**	**0.001**

Table 14: p-values for the statistical significance of differences in performance among the RC components across all data sets. For each cell, bold text means that the first method was better than the second method and italic text means it was worse.

	sample ratio									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	**overall**
wvRN v cdRN	**1.483**	**1.092**	**1.059**	**1.070**	**1.042**	**1.058**	**1.047**	**1.057**	**1.040**	**1.068**
wvRN v nLB	∞	**7.741**	**1.901**	**1.279**	*1.027*	*1.091*	*1.081*	*1.082*	*1.067*	**1.163**
cdRN v nLB	∞	**7.086**	**1.794**	**1.195**	*1.071*	*1.154*	*1.132*	*1.144*	*1.110*	**1.089**

Table 15: Relative error reduction (ER^*_{REL}) improvements for each RC component across all data sets. Each cell shows the ratio of the better method's error reduction over the other method's error reduction. The last column, **overall**, is based on taking the ratio of the average error reduction for the methods across all sample ratios. Bold text means the first method is better and italics means the second method is better. ∞ means that the second method performed worse than the base error.

	sample ratio									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	**total**
wvRN	7	4	4	6	4	4	2	1	2	34
cdRN	5	8	6	2	1	0	0	1	1	24
nLB	0	0	2	4	7	8	10	10	9	50

Table 16: Number of times each RC implementation was the best across the 12 data sets.

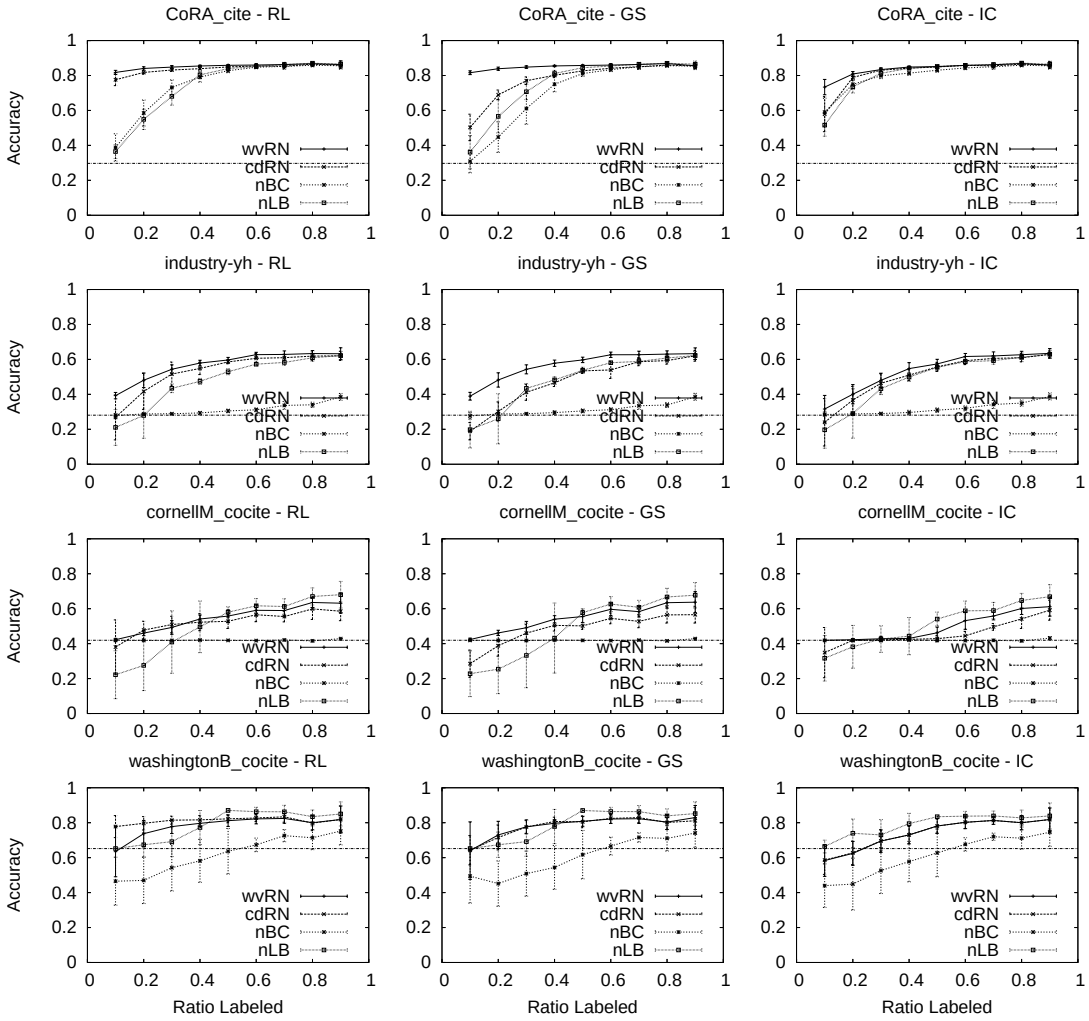

Figure 3: Comparison of Relational Classifiers on a select few data sets. The data set (and link-type) and the paired collective inference component is listed above each graph. The horizontal line represents predicting the most prevalent class.

Table 14 shows statistical significance results, computed as described in the previous section (except here varying the RC component). Clearly, nBC was always significantly worse than the other three RCs and is therefore eliminated from the remainder of this analysis. wvRN was always significantly better than cdRN. Examining the two RN methods versus nLB we see the same pattern: at $r = 0.5$, the advantage shifts from the RN methods to nLB.

Table 15 shows the error reduction ratios for each RC comparison, computed as in the previous section with the obvious changes between RC and CI. The same patterns are evident as observed from Table 14. Further, we see that the differences can be large: when the RN methods are better, they often are much better. The link-based classifier also can be considerably better than wvRN—however, we should keep in mind that wvRN does not learn!

	sample ratio									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	**total**
wvRN-IC	0	0	0	0	0	0	0	0	1	1
wvRN-GS	1	1	0	1	3	0	0	0	0	6
wvRN-RL	6	3	4	5	1	4	2	1	1	27
cdRN-IC	0	0	0	0	0	0	0	0	0	0
cdRN-GS	0	0	0	0	0	0	0	0	0	0
cdRN-RL	5	8	6	2	1	0	0	1	1	24
nBC-IC	0	0	0	0	0	0	0	0	0	0
nBC-GS	0	0	0	0	0	0	0	0	0	0
nBC-RL	0	0	0	0	0	0	0	0	0	0
nLB-IC	0	0	2	4	3	4	2	4	1	20
nLB-GS	0	0	0	0	2	3	4	4	4	17
nLB-RL	0	0	0	0	2	1	4	2	4	13

Table 17: Number of times each RC-CI configuration was the best across the 12 data sets.

	sample ratio								
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90
wvRN-RL v cdRN-RL	*0.400*	*0.002*	*0.400*	*0.400*	**0.999**	**0.300**	**0.100**	**0.005**	**0.001**
wvRN-RL v nLB-IC	**0.001**	**0.001**	**0.001**	**0.100**	*0.050*	*0.001*	*0.001*	*0.001*	*0.001*
wvRN-RL v nLB-GS	**0.001**	**0.001**	**0.001**	**0.001**	**0.100**	*0.050*	*0.005*	*0.001*	*0.001*
wvRN-RL v nLB-RL	**0.001**	**0.001**	**0.001**	**0.001**	**0.200**	*0.001*	*0.001*	*0.001*	*0.001*
cdRN-RL v nLB-IC	**0.001**	**0.001**	**0.001**	**0.050**	*0.050*	*0.001*	*0.001*	*0.001*	*0.001*
cdRN-RL v nLB-GS	**0.001**	**0.001**	**0.001**	**0.001**	**0.100**	*0.020*	*0.001*	*0.001*	*0.001*
cdRN-RL v nLB-RL	**0.001**	**0.001**	**0.001**	**0.001**	**0.200**	*0.001*	*0.001*	*0.001*	*0.001*
nLB-IC v nLB-GS	**0.001**	**0.001**	**0.001**	**0.001**	**0.025**	**0.200**	**0.300**	*0.100*	*0.200*
nLB-IC v nLB-RL	**0.001**	**0.001**	**0.001**	**0.001**	**0.025**	*0.999*	*0.999*	*0.050*	*0.020*
nLB-RL v nLB-GS	**0.999**	**0.050**	**0.250**	**0.025**	**0.300**	**0.100**	**0.200**	**0.300**	**0.050**

Table 18: Statistical significance of differences in performance among the four best RC-CI configurations across all data sets. For each cell, normal text means that the first method was better than the second method and italic text means it was worse.

Table 16 shows how often each RC method participated in the best combination, as described in the previous section. nLB is the overall winner, but we see the same clear pattern that the RN methods dominate for fewer labels, and nLB dominates for more labels, with the advantage changing hands at $r = 0.5$.

3.3.4 INTERACTION BETWEEN COMPONENTS

Table 17 shows how many times each of the twelve individual RC-CI configurations was the best, across the twelve data sets and nine labeling ratios. Five configurations stand out: wvRN-RL, cdRN-RL, and nLB with any of the CI methods. Table 18 and Table 19 compare these five methods analogously to the previous sections. (Here, each cell comprises 120 data points gathered from the 12 data sets times 10 runs.) The clear pattern is in line with that shown in the prior sections, showing that of this set of best methods, the RN-based methods excel for fewer labeled data, and the nLB-based methods excel for more labeled data.

	sample ratio									
	0.10	0.20	0.30	0.40	0.50	0.60	0.70	0.80	0.90	**overall**
wvRN-RL v cdRN-RL	*1.233*	*1.165*	*1.042*	*1.025*	*1.009*	*1.002*	**1.003**	**1.020**	**1.027**	*1.028*
wvRN-RL v nLB-IC	**4.120**	**1.937**	**1.208**	**1.045**	*1.041*	*1.070*	*1.066*	*1.068*	*1.057*	**1.070**
wvRN-RL v nLB-GS	∞	∞	**3.659**	**1.724**	**1.052**	*1.054*	*1.062*	*1.076*	*1.063*	**1.370**
wvRN-RL v nLB-RL	∞	∞	**3.390**	**1.573**	**1.031**	*1.074*	*1.070*	*1.077*	*1.068*	**1.331**
cdRN-RL v nLB-IC	**5.081**	**2.257**	**1.259**	**1.071**	*1.032*	*1.068*	*1.069*	*1.090*	*1.086*	**1.100**
cdRN-RL v nLB-GS	∞	∞	**3.813**	**1.767**	**1.061**	*1.052*	*1.065*	*1.098*	*1.092*	**1.409**
cdRN-RL v nLB-RL	∞	∞	**3.533**	**1.612**	**1.040**	*1.072*	*1.074*	*1.098*	*1.096*	**1.369**
nLB-IC v nLB-GS	∞	∞	**3.028**	**1.649**	**1.095**	**1.015**	**1.004**	*1.007*	*1.005*	**1.281**
nLB-IC v nLB-RL	∞	∞	**2.805**	**1.505**	**1.074**	*1.004*	*1.004*	*1.008*	*1.010*	**1.245**
nLB-RL v nLB-GS	NA	NA	**1.079**	**1.096**	**1.020**	**1.019**	**1.008**	**1.001**	**1.004**	**1.029**

Table 19: Relative error reduction (ER^*_{REL}) improvements for the 5 best RC-CI configurations across all data sets. Each cell shows the ratio of the better method's error reduction over the other method's error reduction. The last column, overall, is based on taking the ratio of the average error reduction for the methods across all sample ratios. Bold text means the first method was better and italics menas the second method was better. ∞ means that the second method performed worse than the base error, and a value of "NA" indicates that both performed worse than the base error.)

In addition, these results show that the RN-methods clearly should be paired with RL. nLB, on the other hand, does not favor one CI method over the others. One possible explanation for the superior performance of the RN/RL combinations is that RL simply performs better with fewer known labels, where propagating uncertainty may be especially worthwhile as compared to working with estimated labelings. However, this does not hold for nLB (where, as more labels are known, RL performs comparably better than IC or GS). Therefore, there must be a more subtle interaction between the RN methods and the CI methods. This remains to be explained.

Following up on these results, a 2-way ANOVA shows a strong interaction between RC and CI components for most data sets for small numbers of labeled nodes, as would be expected given the strong performance of the specific pairings wvRN-RL and cdRN-RL. As more nodes are labeled, the interaction becomes insignificant for almost all data sets, as might be expected given that nLB dominates but no CI component does. The ANOVA suggests that for very many known labels, it matters little which CI method is used.

3.3.5 WHEN THINGS GO WRONG

To create homogeneous graphs, we had to select the edges to use. As mentioned briefly above, the type of edge selected can have a substantial impact on classification accuracy. For these data sets, the worst case (we have found) occurs for WebKB. As described in Section 3.1.3, for the results presented so far we have used co-citation links, based on observations in prior published work. An obvious alternative is to use the hyperlinks themselves.

Figures 4 and 5 show the results of using hyperlinks instead of co-citation links. The network-classification methods perform much worse than in the previous experiments. Although there is some lift at large values of r, especially for the Washington data, the performance is not compara-

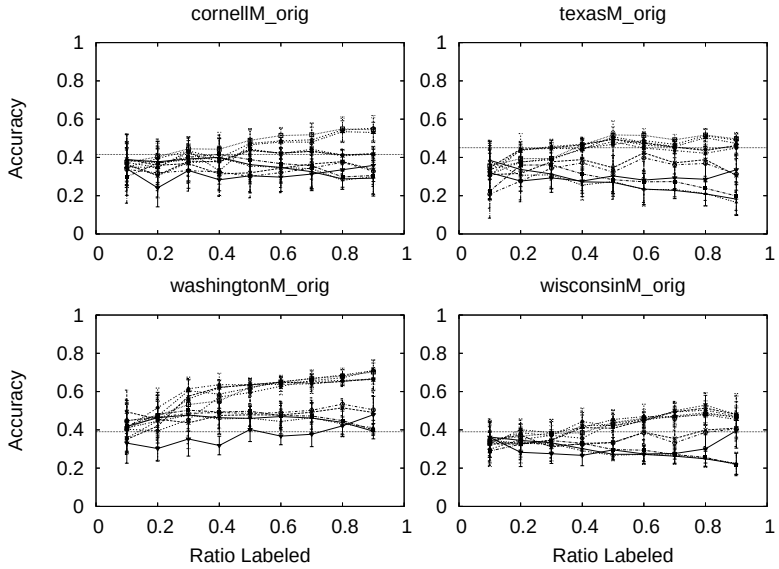

Figure 4: Performances on WebKB multi-class problems using hyperlinks as edges.

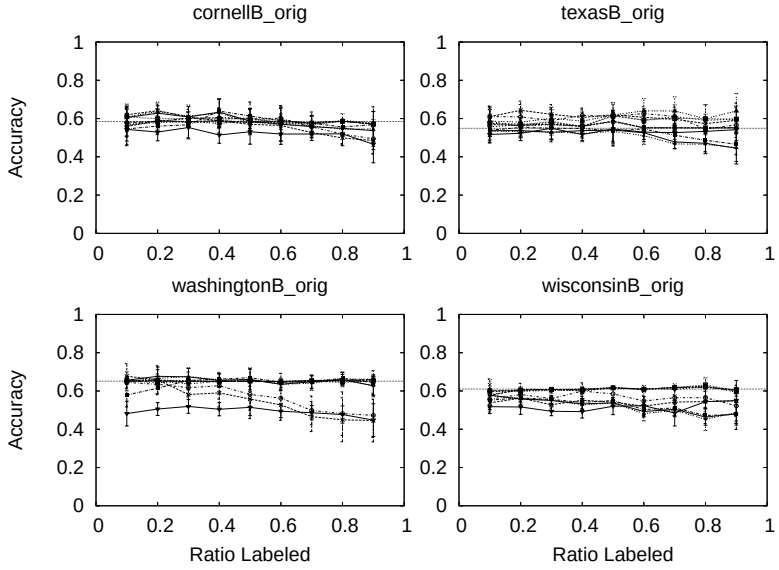

Figure 5: Performances on WebKB binary-class problems using hyperlinks as edges.

ble to that with the co-citation formulation. The transformation from the hyperlink-based network to the co-citation-based network adds no new information to the graph. However, in the hyperlink formulation the network classification methods cannot take full advantage of the information present—mainly because of the first-order Markov assumption made by the relational classifiers. These results demonstrate that the choice of edges can be crucial for good performance.

3.3.6 SELECTING EDGES

Creating a graph with a single type of edge from a problem where various possible links exist is a representation engineering problem reminiscent of the selection of a small set of useful features for traditional classification.[19] For feature selection, practitioners use a combination of domain knowledge and trial and error to select a good representation. To create the networked data for our study, we chose edges based on suggestions from prior work—which indirectly combines domain knowledge and prior trial and error, although we explicitly avoided choosing the representations based on their performance using NetKit.

Pursuing the analogy with choosing features, it may be possible to select edges automatically. It is beyond the scope of this paper to address the general (open and important) problem of edge selection; however, the excellence (on these data sets) and simplicity of wvRN suggests straightforward techniques.

If we consider the data sets used in the study, all but the industry classification data sets have more than one type of edge:

1. **cora**: We linked entities through citations (*cite*). Alternatively, we could have linked by the sharing of an author (*author*), or by either relationship (combined as a single generic link).

2. **imdb**: There are many types of ways to connect two movies, but we focus here on four that were suggested to us by David Jensen: *actor*, *director*, *producer* and production company (*prodco*). Again, we could use any or all of them (we do not consider all possible combinations here).

3. **WebKB**: Based on prior work, we chose co-citations (*cocite*) for the case study and later showed that the original hyperlinks (*hyper*) were a poor choice.

Kohavi and John (1997) differentiate between wrapper approaches and filter approaches to feature selection, and this notion extends directly to edge selection. For any network classification method we can take a wrapper approach, computing the error reduction over G^K using cross-validation. wvRN is an attractive candidate for such an approach, because it is very efficient and requires no training; we can use a simple leave-one-out (loo) estimation.

The homophily-based wvRN also lends itself to a filter approach, selecting the edge type simply by measuring the homophily in G^K. Heckathorn and Jeffi (2003) define a homophily index, but it computes homophily for a specific group, or class, rather than a general value across all classes. The *assortativity coefficient* (Newman, 2003) is based on the correlation between the classes linked by edges in a graph. Specifically, it is based on the graph's assortativity matrix—a CxC matrix, where cell e_{ij} represents the fraction of (all) edges that link nodes of class c_i to nodes of class c_j, such that

19. We required a single edge type for our homogeneous case study; it is reasonable to conjecture that even if heterogeneous links are allowed, a small set of good links would be preferable. For example, a link-based classifier produces a feature vector representation with multiple positions per link type.

Data set	size	base acc.	num edges	mean edge weight	mean node degree	Assortativity (edge) A_E	Assortativity (node) A_N	ER_{REL} loo (wvRN) $r = 0.90$	ER_{REL} wvRN at $r = 0.90$
$\text{cora}_{\text{cite}}$	3583	0.297	22516	2.061	6.284	**0.737**	0.642	0.5373	0.805
cora_{all}	4025	0.315	71824	2.418	17.844	0.656	**0.656**	**0.6122**	0.767
$\text{cora}_{\text{author}}$	3604	0.317	56268	2.262	15.613	0.623	0.558	0.4662	0.711
$\text{imdb}_{\text{prodco}}$	1169	0.511	40634	1.077	34.760	0.501	**0.392**	**0.3711**	0.647
$\text{imdb}_{\text{producers}}$	1195	0.520	13148	1.598	11.003	0.283	0.389	0.3618	0.547
imdb_{all}	1377	0.564	92248	1.307	66.992	0.279	0.308	0.3415	0.531
$\text{imdb}_{\text{directors}}$	554	0.549	826	1.031	1.491	**0.503**	0.210	0.0369	0.498
$\text{imdb}_{\text{actors}}$	1285	0.541	48354	1.135	37.630	0.131	0.174	0.1372	0.246
$\text{cornellB}_{\text{all}}$	349	0.585	27539	3.000	78.908	0.325	**0.399**	**0.5655**	0.629
$\text{cornellB}_{\text{cocite}}$	346	0.581	26832	2.974	77.549	**0.360**	0.394	0.5345	0.618
$\text{cornellB}_{\text{hyper}}$	349	0.585	1393	2.349	3.991	−0.169	−0.068	−0.1621	−0.114
$\text{cornellM}_{\text{all}}$	349	0.415	27539	3.000	78.908	0.219	**0.286**	**0.3209**	0.382
$\text{cornellM}_{\text{cocite}}$	346	0.419	26832	2.974	77.549	**0.227**	0.273	0.2481	0.366
$\text{cornellM}_{\text{hyper}}$	349	0.415	1393	2.349	3.991	0.054	0.102	−0.2883	−0.212
$\text{texasB}_{\text{cocite}}$	334	0.512	32988	2.961	98.766	**0.577**	**0.617**	**0.7166**	0.819
$\text{texasB}_{\text{all}}$	338	0.518	33364	2.995	98.710	0.523	0.585	0.6939	0.768
$\text{texasB}_{\text{hyper}}$	285	0.547	1001	2.605	3.512	−0.179	−0.114	−0.1368	−0.232
$\text{texasM}_{\text{cocite}}$	334	0.488	32988	2.961	98.766	**0.461**	**0.477**	0.3737	0.475
$\text{texasM}_{\text{all}}$	338	0.482	33364	2.995	98.710	0.420	0.458	**0.3874**	0.466
$\text{texasM}_{\text{hyper}}$	285	0.453	1001	2.605	3.512	−0.033	−0.044	−0.6583	−0.490
$\text{washingtonB}_{\text{all}}$	434	0.652	31253	3.800	72.012	**0.388**	**0.455**	**0.4225**	0.530
$\text{washingtonB}_{\text{cocite}}$	434	0.652	30462	3.773	70.189	0.375	0.446	0.3940	0.477
$\text{washingtonB}_{\text{hyper}}$	433	0.651	1941	2.374	4.483	−0.095	0.076	−0.1126	−0.069
$\text{washingtonM}_{\text{cocite}}$	434	0.392	30462	3.773	70.189	0.301	0.359	0.3481	0.503
$\text{washingtonM}_{\text{all}}$	434	0.392	31253	3.800	72.012	**0.331**	**0.377**	**0.4023**	0.453
$\text{washingtonM}_{\text{hyper}}$	433	0.390	1941	2.374	4.483	0.084	0.233	−0.0167	0.004
$\text{wisconsinB}_{\text{all}}$	352	0.560	33587	3.543	95.418	0.524	**0.587**	**0.7219**	0.855
$\text{wisconsinB}_{\text{cocite}}$	348	0.555	33250	3.499	95.546	**0.673**	0.585	0.7168	0.788
$\text{wisconsinB}_{\text{hyper}}$	297	0.616	1152	2.500	3.879	−0.147	−0.103	−0.2123	−0.331
$\text{wisconsinM}_{\text{cocite}}$	348	0.445	33250	3.499	95.546	**0.577**	**0.489**	0.4286	0.544
$\text{wisconsinM}_{\text{all}}$	352	0.440	33587	3.543	95.418	0.416	0.474	**0.4518**	0.503
$\text{wisconsinM}_{\text{hyper}}$	297	0.384	1152	2.500	3.879	0.160	0.021	−0.4729	−0.275
# mistakes						5	2	4	

Table 20: Assortativity details on data sets across various edge types. Each data set grouping is sorted on ER_{REL}. A_E, A_N and ER_{REL} values were all averaged over the 10 data splits used throughout the case study. The leave-one-out measure used only G^K to calculate the ER_{REL} value.

$\sum_{ij} e_{ij} = 1$. The assortativity coefficient, A_E, is calculated as follows:

$$a_i = \sum_j e_{ij} \tag{18}$$

$$b_j = \sum_i e_{ij} \tag{19}$$

$$A_E = \frac{\sum_i e_{ii} - \sum_i a_i \cdot b_i}{1 - \sum_i a_i \cdot b_i} \tag{20}$$

A_E measures homophily across edges, while wvRN is based on homophily across nodes. It is possible to create (sometimes weird) graphs with high A_E but for which wvRN performs poorly, and vice versa. However, we can modify A_E to be a *node-based assortativity* coefficient, A_N, by defining e^*_{ij}, a node-based cell-value in the assortativity matrix as follows:

$$e^*_{ij} = \frac{1}{Z}\text{RV}(X_i)_j, \tag{21}$$

where $\text{RV}(X_i)_j$ is the j^{th} element in $\text{RV}(X_i)$ as defined in Equation 4, and Z is a normalizing constant such that all e_{ij} sum to 1.

To assess their value for edge selection for wvRN, we compute the error reduction for each different edge type (and all edges) for the benchmark data sets, and compare the best with that of the edge selected by each of these three methods (loo, A_E, A_N). In Table 20 the first six columns show the data set, the number of nodes, the base accuracy, the number of edges, the average edge weight, and the average node degree. The next columns show A_E and A_N. The next column shows the estimated ER_{REL} value based on the leave-one-out estimation, and the last column shows the ER_{REL} values on the test set. Each data set group is sorted by the ER_{REL} performance on its various edge types, so the top row is the "best" edge selection. Note that as the edge types differ, we get different connectivities and different coverages, and hence different the values are not completely comparable.

The results show that the links used in our study generally resulted in the highest node-based assortativity.[20] A_N in 8 out of 10 cases chose the best edge. In the two cases where this was not the case, the differences in ER_{REL} were small. Neither the leave-one-out (loo) method nor A_E performed as well, but they nevertheless yield networks on which wvRN performs relatively well. Notice that for IMDb, although director has the highest A_E, it also has very low coverage (only 554 nodes were connected), and with such a slight difference in assortativity between that and prodco there should be no question which should be used for classification. A_N and the leave-one-out estimates are much more volatile than A_E as the amount of labeled data decreases, because there typically are many more edges than nodes. If we believe that assortativity is relatively stable across the network, it may be beneficial to use A_E when little is known. However, for our data sets, A_N performs just as well as A_E even when $r = 0.1$.

3.4 The Case for Network-Only Baseline Methods

On the benchmark data sets, error rates were reduced markedly by taking into account only the class-linkage structure of the network. This argues strongly for using simple, network-only models

20. We had picked the edge types for the study before performing this analysis. However, common sense and domain knowledge lead one to conclude that the edge types we used in the case study would have high assortativity.

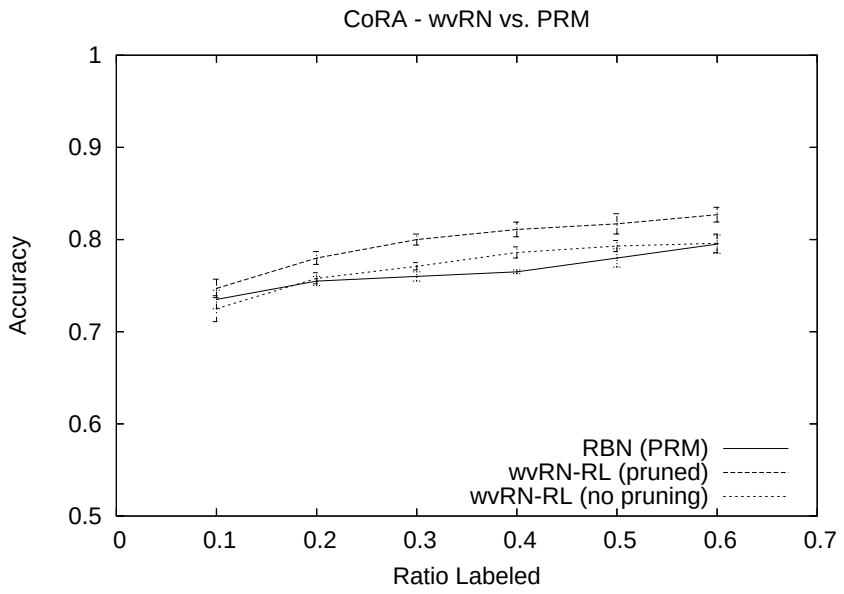

Figure 6: Comparison of wvRN to RBN (PRM) (Taskar et al., 2001). The graph shows wvRN using both citation and author links as in the original study. The "pruned" results follow the methodology of the case study in this paper by removing zero-knowledge components and singletons from the test set.

as baselines in studies of more complex methods for classification in networked data. For example, consider CoRA. In a prior study, Taskar et al. (2001) show that a relational Bayesian network (RBN), there called a Probabilistic Relational Model (PRM), was able to achieve a higher accuracy than a non-relational naive Bayesian classifier for $r = \{0.1, \ldots, 0.6\}$. However, as we saw above, the no-learning wvRN performed quite well on this data set. Figure 6 compares the accuracies of the RBN (transcribed from the graphs in the paper) with wvRN. We see clearly that wvRN was able to perform comparably.[21] This demonstrates that CoRA is not a good data set to showcase the advantages of RBNs for classification. Had a method such as wvRN been readily available as a baseline, then Taskar et al. would most likely have used a more appropriate data set.

More generally, this study has not demonstrated that these benchmark data sets hold little value for studying within-network learning. However, wvRN does set a high bar for studying more-complicated methods for learning classification models for these data sets.

3.5 Limitations

As mentioned earlier, we would like to be able to characterize how much classification ability comes from the structure of the network alone. We have examined a limited notion of using the structure of the network. These methods all assume that "the power of the network" can be reduced to "the power of the neighborhood," bolstered by collective inference, rather than using relational models

21. The "pruned" results show the accuracy after eliminating the zero-knowledge components, for which wvRN can only predict the most prevalent class.

that look deeper. Furthermore, we only considered links and class labels—we did not consider identifying the individual nodes. Networked data allow the identities of particular related entities to be used directly in classification and learning—being linked to Mohammed Atta is informative (Perlich and Provost, 2004).

In the homogeneous, univariate case study we have ignored much of the complexity of real networked data, such as heterogeneous edges, heterogeneous nodes, directed edges, and attributes of nodes and edges. Each of these introduces complications and opportunities for modeling. There are too few comprehensive machine learning studies that consider these dimensions systematically. For example, when using attributes of nodes, how much is gained by using them in the relational classifier, as opposed to using them simply to initialize priors? (For example, Chakrabarti et al. (1998) found that using the text of hyperlinked documents reduced performance.) Similarly, how much value is added by considering multiple edge types explicitly?

An important limitation of this work, with respect to its relevance to practical problems, is that we randomly choose training data to be labeled. It is likely that the data for which labels are available are interdependent. For example, all the members from one terrorist cell may be known and none from another. If other attributes are available more uniformly, then studies such as this may artificially favor network-only methods over attribute-based methods.

Another limitation of this study is that we have not completely explained the very poor performance of nBC (used by Chakrabarti et al. (1998)). Our treatment of zeros does not seem to be the culprit; for example, zeros are rare in the binary classification problems. As with naive Bayes more generally, the posterior estimates typically are extreme and this is exacerbated by having many neighbors (as one would expect if the independence assumption is grossly violated). Poorly calibrated probability estimates are problematic for the collective inference methods–for example, consider Gibbs sampling given posteriors comprising essentially zeros and ones. We are investigating this further.

3.6 Conclusions and Future Work

We introduced a modular toolkit, NetKit-SRL, for classification in networked data. The importance of NetKit is three-fold: (1) it generalizes several existing methods for classification in networked data, thereby making comparison to existing methods possible; (2) it enables the creation and use of many new algorithms by its modularity and extensibility, for example as demonstrated with nLB-GS, nLB-RL, and cdRN-RL, which were among the five best network classifiers in the case study, and (3) it enables the analysis/comparison of individual components and configurations.

We used NetKit to perform a case study of within-network, univariate classification for homogeneous networked data. The case study makes several contributions. It provides demonstrative support for points 2 and 3 above. By comparing the various components and combinations, clear patterns appear. Certain collective inference and relational classification components stand out with consistently better performance: for CI, relaxation labeling was best; for RC, the link-based classifier was clearly preferable when many labels were known. The lower-variance methods (wvRN and cdRN) dominated when fewer labels were known. In combination, five RC-CI methods stand out strikingly: nLB with one of the CI methods dominates when many labels are known; wvRN-RL and cdRN-RL dominate when fewer labels are known.

More generally, the results showcase two different modes of within-network classification: cases when many labels are known ex ante versus cases where few are known. The former scenario may

correspond (for example) to networks that evolve over time with new nodes needing classification, as would be the case for predicting movie box-office receipts. Examples of the little-known scenario can be found in counter-terrorism and law enforcement, where analysts form complex interaction networks containing a few, known bad guys. The little-known scenario has an economic component, similar to active learning: it may be worthwhile to incur costs to label additional nodes in the network, because this will lead to much improved classification. This suggests another direction for future work—identifying the most beneficial nodes for labeling (cf., Domingos and Richardson (2001)).

The case study also showcases a problem of representation for network classification: the selection of which edges to use. It is straightforward to extend NetKit's RC methods to handle heterogeneous links. However, that would not solve the fundamental problem that edge selection, like feature selection for traditional learning, may improve generalization performance (as well as provide simpler models).

Finally, the case study demonstrated the power of simple network classification models. On the benchmark data sets, error rates were reduced markedly by taking into account only the class-linkage structure of the network. No attribute information was used. Although learning helped in many cases, the no-learning wvRN was a very strong competitor—performing very well when few labels were known. This argues strongly for using simple, network-only models as baselines in studies of classification in networked data. It also calls raises the question of whether we need more powerful methods or "better" benchmark data sets.

Classification in networked data is important for real-world applications, and presents many opportunities for machine-learning research. The field is beginning to amass benchmark domains containing networked data. We hope that NetKit can facilitate systematic study.

Acknowledgments

David Jensen made many helpful suggestions, including pointing us to the WebKB data set and suggesting ways to do well on it. Abraham Bernstein collaborated on the generation of the two industry classification data sets. Kaveh R. Ghazi worked diligently coding the open-source NetKit and ensuring that it would be ready for release on schedule. We thank Ben Taskar and Andrew McCallum for providing us with versions of the Cora data set. Thanks to Sugato Basu, Misha Bilenko, Pedro Domingos, Joydeep Ghosh, David Jensen, Andrew McCallum, Jennifer Neville, Mark Newman, Claudia Perlich, and audience members of talks for helpful comments and discussions.

References

J. C. Almack. The Influence of Intelligence on the Selection of Associates. *School and Society*, 16: 529–530, 1922.

A. Bernstein, S. Clearwater, S. Hill, C. Perlich, and F. Provost. Discovering Knowledge from Relational Data Extracted from Business News. In *Proceedings of the Multi-Relational Data Mining Workshop (MRDM) at the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2002.

A. Bernstein, S. Clearwater, and F. Provost. The Relational Vector-space Model and Industry Classification. In *Proceedings of the Learning Statistical Models from Relational Data Workshop*

(SRL) at the 19th International Joint Conference on Artificial Intelligence (IJCAI), pages 8–18, 2003.

P. M. Blau. *Inequality and Heterogeneity: A Primitive Theory of Social Structure*. New York: Free Press, 1977.

H. Bott. Observation of Play Activities in a Nursery School. *Genetic Psychology Monographs*, 4: 44–88, 1928.

Y. Boykov, O. Veksler, and R. Zabih. Fast Approximate Energy Minimization via Graph Cuts. *IEEE Transactions on Pattern Analysis and Machine Intellingence (PAMI)*, 23(11), November 2001.

S. Chakrabarti, B. Dom, and P. Indyk. Enhanced Hypertext Categorization Using Hyperlinks. In *ACM SIGMOD International Conference on Management of Data*, 1998.

C. Cortes, D. Pregibon, and C. T. Volinsky. Communities of Interest. In *Proceedings of Intelligent Data Analysis (IDA)*, 2001.

M. Craven, D. Freitag, A. McCallum, T. Mitchell, K. Nigam, and C. Y. Quek. Learning to Extract Symbolic Knowledge from the World Wide Web. In *15th Conference of the American Association for Artificial Intelligence*, 1998.

R. L. Dobrushin. The Description of a Random Field by Means of Conditional Probabilities and Conditions of its Regularity. *Theory of Probability and its Application*, 13(2):197–224, 1968.

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

P. Domingos and M. Richardson. Markov Logic: A Unifying Framework for Statistical Relational Learning. In *Proceedings of the ICML-2004 Workshop on Statistical Relational Learning and its Connections to Other Fields*, pages 49–54, 2004.

S. Dzeroski and N. Lavrac. *Relational Data Mining*. Berlin; New York: Springer, 2001.

W. Emde and D. Wettschereck. Relational Instance-Based Learning. In Lorenza Saitta, editor, *Proceedings of the 13th International Conference on Machine Learning (ICML)*, pages 122–130. Morgan Kaufmann, 1996.

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

T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proceedings of the Fifth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 1999.

P. A. Flach and N. Lachiche. 1BC: A First-Order Bayesian Classifier. In Saso Dzeroski and Peter A. Flach, editors, *Proceedings of the Ninth International Workshop on Inductive Logic Programming (ILP)*, volume 1634, pages 92–103. Springer-Verlag, June 1999.

N. Friedman, L. Getoor, D. Koller, and A. Pfeffer. Learning Probabilistic Relational Models. In *Sixteenth International Joint Conference on Artificial Intelligence (IJCAI)*, 1999.

S. Geman and D. Geman. Stochastic relaxation, Gibbs distributions and the Bayesian restoration of images. *IEEE Transactions on Pattern Analysis and Machine Intellingence (PAMI)*, 6:721–741, 1984.

W. R. Gilks, S. Richardson, and D. J. Spiegelhalter. *Markov Chain Monte Carlo in Practice*. Chapman & Hall/CRC, 1995.

D. D. Heckathorn and J. Jeffi. Jazz networks: Using respondent-driven sampling to study stratification in two jazz musician communities. In *American Sociological Association meetings*, August 2003.

D. Heckerman, D. M. Chickering, C. Meek, R. Rounthwaite, and C .Kadie. Dependency Networks for Inference, Collaborative Filtering, and Data Visualization. *Journal of Machine Learning Research (JMLR)*, 1:49–75, October 2000.

D. Jensen and J .Neville. Data Mining in Social Networks. In *National Academy of Sciences workshop on Dynamic Social Network Modeling and Analysis*, 2002a.

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 (ICML)*, 2002b.

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*, 2004.

R. Kohavi and G. John. Wrappers for Feature Subset Selection. *Artificial Intelligence special issue on Relevance*, 97(1–2):273–324, 1997.

D. Koller and A. Pfeffer. Probabilistic Frame-Based Systems. In *AAAI/IAAI*, pages 580–587, 1998.

S. Kramer, N. Lavrac, and P. Flach. Propositionalization approaches to relational data mining. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 262–291. Springer-Verlag, 2001.

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 (ICML)*, 2001.

P. Lazarsfeld and R. K. Merton. Friendship as a Social Process: A Substantive and Methodological Analysis. In Morroe Berger, Theodore Abel, and Charles H. Page, editors, *Freedom and Control in Modern Society*, pages 18–66. Van Nostrand, 1954.

C. P. Loomis. Political and Occupational Cleavages in a Hanoverian Village. *Sociometry*, 9:316–3333, 1946.

Q. Lu and L. Getoor. Link-Based Classification. In *Proceedings of the 20th International Conference on Machine Learning (ICML)*, 2003.

S. A. Macskassy and F. Provost. A Simple Relational Classifier. In *Proceedings of the Multi-Relational Data Mining Workshop (MRDM) at the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

A. McCallum, K. Nigam, J. Rennie, and K. Seymore. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval*, 3(2):127–163, 2000.

M. McPherson, L. Smith-Lovin, and J. M. Cook. Birds of a Feather: Homophily in Social Networks. *Annual Review of Sociology*, 27:415–444, 2001.

K. Murphy, Y. Weiss, and M. I. Jordan. Loopy Belief-propagation for Approximate Inference: An Empirical Study. In K. B. Laskey and H. Prade, editors, *Proceedings of the Fifteenth Conference on Uncertainty in Artificial Intelligence (UAI)*. Morgan Kaufmann, 1999.

J. Neville and D. Jensen. Iterative Classification in Relational Data. In *AAAI Workshop on Learning Statistical Models from Relational Data*, pages 13–20, 2000.

J. Neville and D. Jensen. Collective Classification with Relational Dependency Networks. In *Proceedings of the Second Workshop on Multi-Relational Data Mining (MRDM-2003) at KDD-2003*, 2003.

J. Neville and D. Jensen. Dependency Networks for Relational Data. In *Proceedings of the Fourth IEEE International Conference in Data Mining (ICDM)*, 2004.

J. Neville, D. Jensen, L. Friedland, and M. Hay. Learning Relational Probability Trees. In *Proceedings of the Ninth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2003.

M.. J. Newman. Mixing patterns in networks. *Physical Review E*, 67, 2003. 026126.

J. Pearl. *Probabilistic Reasoning in Intelligent Systems*. Morgan Kaufmann, 1988.

C. Perlich. Citation-Based Document Classification. In *Workshop on Information Technology and Systems (WITS)*, 2003.

C. Perlich and F. Provost. Aggregation-based feature invention and relational concept classes. In *KDD*, 2003.

C. Perlich and F. Provost. ACORA: Distribution-based Aggregation for Relational Learning from Identifier Attributes. Technical Report CeDER Working Paper CeDER-04-04, Stern School of Business, New York University, 2004.

A. Popescul and L. H. Ungar. Statistical Relational Learning for Link Prediction. In *Proceedings of the Learning Statistical Models from Relational Data Workshop (SRL) at the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, 2003.

F. Provost, C. Perlich, and S. A. Macskassy. Relational Learning Problems and Simple Models. In *Proceedings of the Learning Statistical Models from Relational Data Workshop (SRL) at the 19th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 116–120, 2003.

L. De Raedt, H. Blockeel, L. Dehaspe, and W. Van Laer. Three companions for data mining in first order logic. In Saso Dzeroski and Nada Lavrac, editors, *Relational Data Mining*, pages 105–139. Springer-Verlag, 2001.

H. M. Richardson. Community of Values as a Factor in Friendships of College and Adult Women. *Journal of Social Psychology*, 11:303–312, 1940.

J. Rocchio. Relevance feedback in information retrieval. In Salton, editor, *The SMART Retrieval System: Experiments in Automatic Document Processing*, chapter 14, pages 313–323. Prentice–Hall, 1971.

L. J. Savage. *The Foundations of Statistics*. John Wiley and Sons, 1954.

B. Taskar, P. Abbeel, and D. Koller. Discriminative Probabilistic Models for Relational Data. In *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI)*, August 2002.

B. Taskar, E. Segal, and D. Koller. Probabilistic Classification and Clustering in Relational Data. In *Proceedings of the 17th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 870–878, 2001.

G. Winkler. *Image Analysis, Random Fields and Markov Chain Monte Carlo Methods*. Springer-Verlag, 2nd edition, 2003.

I. H. Witten and E. Frank. In *Data Mining: Practical machine learning tools with Java implementations*. Morgan Kaufmann, San Francisco, 2000.

Appendix A. Glossary

cdRN Class Distribution Relational Neighbor Classifier. See Section 2.4.1.

CI Collective Inference Method. See Section 2.4.2.

$\mathcal{D}$ A data set. See Section 3.2.

$\mathcal{D}^K$ What is known about $\mathcal{D}$. See Section 3.2.

$\mathcal{D}^U$ What is not known (and hence what needs to be predicted) about $\mathcal{D}$. See Section 3.2.

GS Gibbs Sampling. See Section 2.4.2.

IC Iterative Classification. See Section 2.4.2.

LC Local Classifier. See Section 2.1.

nBC Network-only Bayes Classifier. See Section 2.4.1.

NC Network-Classification System. An LC-RC-CI combination. See Section 3.2.

nLB Network-only Linked-Based Classifier. See Section 2.4.1.

r The ratio of data which is known in the network. See Section 3.3.1.

RC Relational Classifier. See Section 2.4.1.

RL Relaxation Labeling. See Section 2.4.2.

wvRN Weighted Vote Relational Neighbor Classifier. See Section 2.4.1.

Appendix B. Implementation Notes Regarding NetKit

This section describes in more detail the primary modules.

The current version of NetKit can be obtained from the primary author of this paper. We are currently getting the toolkit ready to be released as open-source (Java 1.5).

B.1 Input Module

This module reads in the given data and represents it as a graph. This module supports heterogeneous edges and nodes although the classification algorithms all assume homogeneous nodes. The edges can be weighted and/or directed.

The data input that the toolkit currently supports consists of a set of flat files, with a schema file defining the overall schema and the files where to read the data from. Each node type and edge type are in separate flat files.

B.2 Local Classifier (LC) Module

The Local Classifier (LC) module is a general application programming interface (API), which enables the implementation of "local" classifiers.

The API consists of two main interface methods: `induceModel(`V^K`)` and `estimate(`v`)`, where v is a vertex in the graph for which to predict the class attribute.

The `induceModel(`V^K`)` methods takes as its input a set of vertices, V^K, and induces an internal model, $\mathcal{M}_L$, which can be used to estimate $P(x|v)$.

The `estimate(`v_i`)` method takes as its input a vertex in the graph and returns a normalized vector, $\mathbf{c}$, where the k-th item, $c(k)$, corresponds to the probability that x takes on the categorical class value $X_k \in \mathcal{X}$.

The toolkit, through the Weka wrapper described below, fully supports the use of any classifiers available in Weka. The toolkit, for experimental purposes, also has the three strawman classifiers which predict a uniform prior, the class prior, or `null`.

Extending NetKit by creating a new local classifier requires one to create a new subclass of the generic NetKit classifier (ClassifierImp) and write a minimum of 5 methods:

1. `public String getShortName()`
2. `public String getName()`
3. `public String getDescription()`
4. `public boolean estimate(Node node, double[] result)`
5. `public void induceModel(Graph g, DataSplit split)`

Once a new class has been created, it must be added to the `lclassifier.properties` configuration file to let NetKit know about its existence.

B.3 Relational Classifier (RC) Module

As with the LC module, the Relational Classifier (RC) module is a general API which enables the implementation of relational classifiers. As with LC, the module consists two main methods: `induceModel(`G^K`)` and `estimate(`v`)`.

The `induceModel(`V^K`)` methods takes as its input the set of vertices, V^K, and induces an internal model, $\mathcal{M}_R$, which can be used to estimate $P(x|v)$.

The `estimate(`v_i`)` method takes as its input the vertex v_i and returns a normalized vector, $\mathbf{c}_i$, where the k-th item, $c_i(k)$, corresponds to the probability that x_i takes on the categorical class value $X_k \in \mathcal{X}$.

The toolkit fully supports the use of any Weka classifiers, which are turned into relational classifiers through the use of aggregation of neighbor attributes.

This module can be configured to aggregate only on the class attribute or on all neighbor attributes. It currently only supports aggregation of direct neighbors. It can further be configured to not make use of intrinsic variables, for experimental studies such as the one performed in this paper.

Extending NetKit by creating a new relational classifier requires one to create a new subclass of the generic NetKit network classifier (NetworkClassifierImp) and write a minimum of 6 methods:

1. public String getShortName()
2. public String getName()
3. public String getDescription()
4. public boolean includeClassAttribute()
5. public boolean doEstimate(Node node, double[] result)
6. public void induceModel(Graph g, DataSplit split)

For ease-of-use, the default implementation has a helper method,

makeVector(Node node, double[] vector),

which takes the intrinsic variables and all the aggregators used by the model and create a vector representation of doubles. This is what is used by the Weka-wrapper module.

Once a new class has been created, it must be added to the `rclassifier.properties` configuration file to let NetKit know about its existence.

B.4 Collective Inferencing Module

The Collective Inferencing (CI) module is a general API which enables the implementation of inferencing techniques. The API consists of one main method: `estimate(`$\mathcal{M}_R$`,` V^U`)`, which takes as its input a learned relational model, $\mathcal{M}_R$, and the set of vertices whose value of attribute x needs to be estimated. It returns $C = \{\mathbf{c}_i\}$.

There are currently three collective inferencing algorithms implemented, each of which are described in Section 2.4.2.

Extending NetKit by creating a new collective inferencing method requires one to create a new subclass of the generic NetKit InferenceMethod class and write a minimum of 4 methods:

1. public String getShortName()
2. public String getName()
3. public String getDescription()
4. public boolean iterate(NetworkClassifier classifier)

This should iterate through the list of nodes whose attributes are to be predicted and apply the classifier to those nodes. How this is done, and what to give the classifier is dependent on the inference method.

Once a new class has been created, it must be added to the `inferencemethod.properties` configuration file to let NetKit know about its existence.

B.5 Aggregators

The toolkit currently supports the more common aggregation techniques, which include the mode, mean, min, max, count, exist and ratio (a normalized count). There are plans to extend these to also include class-conditional aggregation (Perlich and Provost, 2003).

Extending NetKit by creating a new aggregator requires one to either subclass the AggregatorImp class or AggregatorByValueImp class, depending on whether the aggregator is across all values of an attribute (such as min/mode/max) or for a particular attribute value (such as count/exist/ratio.)

Once a new class has been created, it must be added to the `aggregator.properties` configuration file to let NetKit know about its existence.

B.6 Weka Wrapping Module

The final module is the Weka wrapping module. This module acts as a bridge to Weka, a popular public machine learning toolkit. It needs to be initialized by giving it the name of the Weka classifier, WC, to wrap.

There are two wrappers for weka, one for the LC module and one for the RC module, where the `induceModel` and `estimate` methods convert the inputs to the internal representation used by Weka, and then passes this transformed set of entities to WC to let Weka induce the classifier.

The `estimate` method works similarly by converting the attribute vector A into the internal representation used by Weka (again, making use of the aggregator functions specified in the `induceModel` method), calls WC to estimate x_i, and then transforms the reply from WC back into the vector format $\mathbf{c}$ used by our toolkit.

B.7 Configuring NetKit

NetKit is very configurable and should require very little programming for most uses. The configuration files allow great customization of how the LC, RC, and CI modules work, by being able to set many parameters such as how many iterations the CI methods should run for, as well as what kind of aggregation and aggregators the RC methods should use.

There are 7 configuration files:

1. `aggregator.properties`
   This defines the aggregators available as well as what kind of attributes (continuous, categorical, discrete) they will work on.

2. `distance.properties`:
   This defines the vector-distance functions available. Currently, there are the three commonly used distance functions, L1, L2 and cosine. Currently, only one classifier, cdRN, makes use of distance functions.

3. `inferencemethod.properties`:
   This defines, and sets the parameters, for all the inferencemethods available to NetKit. Each method and patemeter specification group is given a unique name such that the same method can be used more than once but with different parameters.

4. `lclassifier.properties`:
   Like the inferencemethod above, this defines and sets the parameters for the local classifiers.

5. `NetKit.properties`:
   This sets default parameters for NetKit (which can be overridden on the commandline.)

6. `rclassifier.properties`:
   Like the inferencemethod above, this defines and sets the parameters for the relational classifiers.

7. `weka.properties`:
   This defines the weka classifiers available to NetKit.

Each of the configuration files are well-commented to make it easy to customize NetKit.