Leveraging Neighbor Attributes for Classification in Sparsely-Labeled Networks

LUKE K. MCDOWELL, U.S. Naval Academy
DAVID W. AHA, Naval Research Laboratory

Many analysis tasks involve linked nodes, such as people connected by friendship links. Research on *link-based classification* (LBC) has studied how to leverage these connections to improve classification accuracy. Most such prior research has assumed the provision of a densely-labeled training network. Instead, this article studies the common and challenging case when LBC must use a single sparsely-labeled network for both learning and inference, a case where existing methods often yield poor accuracy. To address this challenge, we introduce a novel method that enables prediction via "neighbor attributes," which were briefly considered by early LBC work but then abandoned due to perceived problems. We then explain, using both extensive experiments and loss decomposition analysis, how using neighbor attributes often significantly improves accuracy. We further show that using appropriate semi-supervised learning (SSL) is essential to obtaining the best accuracy in this domain, and that the gains of neighbor attributes remain across a range of SSL choices and data conditions. Finally, given the challenges of label sparsity for LBC and the impact of neighbor attributes, we show that multiple previous studies must be re-considered, including studies regarding the best model features, the impact of noisy attributes, and strategies for active learning.

CCS Concepts: •**Information systems** → **Data mining;**

Additional Key Words and Phrases: Collective classification, collective inference, link-based classification, statistical relational learning

ACM Reference Format:
Luke K. McDowell and David W. Aha, 2016. Leveraging Neighbor Attributes for Classification in Sparsely-Labeled Networks. *ACM Trans. Knowl. Discov. Data.* V, N, Article A (January YYYY), 37 pages.
DOI: 0000001.0000001

1. INTRODUCTION

Many problems in communications, social networks, biology, business, and other domains involve classifying nodes in a graph. For instance, consider predicting a class label for each page (node) in a set of linked webpages, where some node labels are provided for learning. A traditional method would use the attributes of each page (e.g., words in the page) to predict its label. In contrast, *link-based classification* (LBC) [Chakrabarti et al. 1998; Neville and Jensen 2000; Taskar et al. 2001] also uses, for each node, the attributes or labels of *neighboring* pages as model features. If "neighbor labels" are used, then an iterative algorithm for *collective inference* is needed, since many labels are initially unknown [Jensen et al. 2004]. If, on the other hand, "neighbor attributes" are used, then a single step of *relational inference* suffices, since typically all attribute values are known.

Link-based classification has been actively studied for over a decade, and continues to attract significant interest in the machine learning [Bilgic et al. 2010; Wang et al. 2011; Saha et al. 2014], data mining [Menon and Elkan 2010; Namata et al. 2011; Jacob et al. 2014], and knowledge management communities [Shi et al. 2011; Kong

This work was supported in part by NSF award number IIS-1116439 and a grant from ONR.
Author's addresses: L. McDowell, Department of Computer Science, U.S. Naval Academy, Annapolis, MD 21402; D. Aha, Navy Center for Applied Research in Artificial Intelligence, Naval Research Laboratory (Code 5514), Washington, DC 20375.

YYYY 1556-4681/YYYY/01-ARTA $15.00
DOI: 0000001.0000001

et al. 2012; Pfeiffer III et al. 2014a]. Despite the additional complexity of inference, recent work has used collective inference (CI) much more frequently than relational inference (RI) for two reasons. First, multiple algorithms for CI (e.g., belief propagation, Gibbs sampling, iterative classification) can substantially increase classification accuracy [Neville and Jensen 2007; Sen et al. 2008]. In contrast, comparisons found RI to be inferior to CI [Jensen et al. 2004] and to sometimes even decrease accuracy compared to methods that ignore links [Chakrabarti et al. 1998]. Second, although RI does not require multiple inference steps, using neighbor attributes as model features is more complex than with neighbor labels, due to the interplay between the larger number of attributes (vs. one label) and a varying number of neighbors for each node. In particular, RI does not naturally mesh with popular, discriminative classifiers such as logistic regression. Because neighbor attributes have appeared to be both difficult to use and unnecessary, very few recent studies have considered them, and none have evaluated how well they fared vs. using neighbor labels.

Most work on link-based classification assumes a fully-labeled training graph. However, while collecting the node attributes and link structure for this graph may be often easy (e.g., for social and webpage networks), acquiring the desired labels can be much more expensive [Gallagher et al. 2008; Neumann et al. 2013]. In response, recent studies have examined CI methods with partially-labeled training graphs, using some semi-supervised learning (SSL) to leverage the unlabeled portion of the graph [Xiang and Neville 2008; Bilgic et al. 2010; Shi et al. 2011; Pfeiffer III et al. 2014b]. However, they have reported weak or inconsistent results, even when using the same datasets and similar algorithms. This includes Bilgic et al., who found moderate gains from SSL, whereas Shi et al. reported otherwise. Likewise, Pfeiffer III et al. reported gains from SSL, but only when using a "composite" likelihood function that limits the extent to which unlabeled nodes are used for learning. Moreover, none of these works evaluated RI.

This article argues that using neighbor attributes as model features can be, contrary to prior belief and practice, an efficient and highly effective method that significantly increases LBC accuracy *when the network is sparsely labeled*. We show that this approach is further enhanced by the use of effective SSL, and that it maintains substantial gains, compared to alternative models, with or without the use of SSL. Furthermore, we argue that including baseline models that use neighbor attributes should be considered *essential* for future experimental studies of LBC, given our results showing their frequent accuracy gains vs. models that ignore such attributes. More specifically, our contributions are as follows:

(1) We provide the first evaluation of LBC that compares models based on neighbor labels (CI) vs. models based on neighbor attributes (RI), for sparsely-labeled networks. Unlike prior studies with fully-labeled training networks, we find that RI is often significantly more accurate than CI.
(2) We introduce an efficient technique, Multi-Neighbor Attribute Classification (MNAC), that enables discriminative classifiers like logistic regression to be used with neighbor attributes, further increasing accuracy.
(3) We examine multiple variants of SSL algorithms. We show for the first time that these methods can improve accuracy with both RI and CI, but that CI is especially sensitive to the specific techniques chosen.
(4) We examine the reasons for RI's gains over CI, and show via a loss decomposition analysis that they arise from differences in learning variance and inference bias.
(5) We demonstrate that these results persist across a wide range of real and synthetic datasets, learning algorithms, inference algorithms, and labeling conditions, even in some situations where CI was thought to be clearly preferable.

(6) Finally, we show that a number of prior studies should be re-evaluated, with sparsely-labeled networks, given our findings regarding the best ways to apply SSL and the surprising usefulness of neighbor attributes for LBC.

The next section defines the LBC problem and introduces CI, RI, and a combination of them, RCI. Section 3 discusses why label sparsity makes LBC so challenging, along with a summary of collective inference. Section 4 explains existing methods for using neighbor attributes, why they have not been frequently used, and then introduces our new MNAC method.[1] Next, Section 5 explains the methods for semi-supervised learning that we consider, and Section 6 describes our experimental method. Section 7 presents experimental results comparing CI, RI, and RCI, where we vary the learning method, label density, and distribution of labels. Section 8 presents additional results that seek to better understand the important differences between RI and CI, while Section 9 describes additional related work. Finally, Section 10 concludes.

2. LINK-BASED CLASSIFICATION

Assume we are given a graph $G = (V, E, X, Y, C)$ where V is a set of nodes, E is a set of edges (links), each $\vec{x_i} \in X$ is an attribute vector for a node $v_i \in V$, each $Y_i \in Y$ is a label variable for v_i, and C is the set of possible labels. We are also given a set of "known" values Y^K for nodes $V^K \subset V$, so that $Y^K = \{y_i | v_i \in V^K\}$. For later convenience, let $\mathcal{N}_i = \{v_j | (i, j) \in E\}$, i.e., the set of nodes in the "neighborhood" of v_i (for simplicity, we assume E is undirected here). Then the *within-network classification task* is to infer Y^U, the values of Y_i for the remaining nodes V^U with "unknown" values ($V^U = V \setminus V^K$).

Running example: Given a (partially-labeled) set of interlinked university webpages, consider the task of predicting whether each page belongs to a professor or a student. There are three kinds of features typically used for this task:

— **Self attributes:** features based on the the textual content of each page (node), e.g., the presence or absence of the word "teaching" for node v.
— **Neighbor attributes:** features based on the attributes of pages that link to v. These may be useful because, e.g., pages often link to others with the same label [Jensen et al. 2004], and neighbor attributes may be used to leverage such correlations.
— **Neighbor labels:** features based on the labels of pages that link to v, such as "Count the number of v's neighbors with the label `Student`."

Table I characterizes classification models based on the kinds of features they use. The simplest, baseline models use only one kind. First, SELFATTRS uses only self attributes. Second, NEIGHATTRS classifies a node v using *only* v's neighbors' attributes. This model has not been previously studied, but we use it to help measure the value of neighbor attributes on their own. Finally, NEIGHLABELS uses only neighbor labels. For instance, the "wvRN" method repeatedly averages the predicted label distributions of a node's neighbors; this performs surprisingly well for some datasets [Macskassy and Provost 2007].

The most popular models combine self attributes with other features. If a model also uses neighbor attributes, then it is performing "relational inference" and we call it RI. A CI model uses neighbor labels instead, via features like the "count `Students`" described above. However, this is challenging, because some labels are unknown and must be estimated, typically with an iterative process of *collective inference* (i.e., CI) [Jensen et al. 2004], which is described further below. Finally, RCI (relational collective inference) uses all three kinds of features, including neighbor labels, so it also must use some kind of collective inference such as Gibbs sampling.

[1]McDowell and Aha [2013] contains an initial report on some parts of this work.

Table I. Types of models, based on the kinds of features used. Some notation is adapted from Jensen et al. [2004].

Model name	Description	Self attr.	Neigh. attr.	Neigh. labels
RCI	Relational collective inference	✓	✓	✓
RI	Relational inference	✓	✓	
CI	Collective inference	✓		✓
SELFATTRS	Self attributes (only)	✓		
NEIGHATTRS	Neighbor attributes (only)		✓	
NEIGHLABELS	Neighbor labels (only)			✓

Table II. Explanation of terms/acronyms used in this article. See also the models in Table I.

Term/acronym	Meaning
	Inference Algorithms
MNAC	Multi-Neighbor Attribute Classification (Section 4.3)
ICA	Iterative Classification Algorithm (Section 2.1)
VMF	Variational Mean Field (Section 2.1, cf., Appendix B.1.2)
Gibbs	Gibbs Sampling (Section 2.1, cf., Appendix B.1.3)
wvRN	Weighted-vote Relat. Neighbor (used for NEIGHLABELS; see Section 2)
	Learning Algorithms/Approaches
SSL	Semi-supervised Learning (Section 5)
SSL-ONCE	One round of SSL for learning (Section 5.2)
SSL-TWICE	Two rounds of SSL for learning (Section 5.2)
SSL-EM	Multiple rounds of SSL (Expectation Maximization) (Section 5.2)
	Other Terms
LBC	Link-based Classification
LR	Logistic regression (classification method)
NB	Naive Bayes (classification method)

2.1. Background on Learning and Inference for LBC

Because most prior work uses CI, we explain learning and inference in this section using CI; later sections consider RCI and RI. Notation is adapted from Pfeiffer III et al. [2014b]. To aid the reader, Table II summarizes many of the terms/acronyms that are introduced in the next few sections.

Learning: The goal of LBC is to jointly infer the missing labels Y^U given the graph's nodes, links, attributes, and known labels, i.e., to compute

$$P(Y^U|V, E, X, Y^K, \theta_M) \tag{1}$$

Performing this inference will require learning the parameters θ_M of a model M that best explains the known, provided labels Y^K given the available evidence. We can do this by maximizing the joint likelihood as follows:

$$\theta_M = \arg\max_{\theta_M} P(Y^K|V, E, X, \theta_M) \tag{2}$$

For simplicity, we omit details of the regularization used to prevent over-fitting; Section 6.2 provides more detail.

This formalization allows for an arbitrary set of dependencies among any of the nodes. However, learning and inference in such a general model is typically intractable for large networks. A very common assumption, therefore, is to approximate the true data likelihood with the *pseudolikelihood* [Besag 1975]. Learning with the pseudolikelihood allows us to use a *local conditional model* or *local classifier* that is not required to factor the full joint distribution, and that makes some conditional independence assumptions that simplify learning and inference. For instance, since CI assumes that the class label of a node v_i depends only on v_i's attributes ($\vec{x_i}$) and on v_i's neighbors' labels ($Y_{\mathcal{N}_i} = \{y_j | v_j \in \mathcal{N}_i\}$), then learning reduces to maximizing:

$$\theta_M = \arg\max_{\theta_M} \sum_{v_i \in V^K} \log P(y_i | \vec{x_i}, Y_{\mathcal{N}_i}, \theta_M) \tag{3}$$

.

Here we are using CI, and $P(y_i|\vec{x_i}, Y_{\mathcal{N}_i}, \theta_M)$ is the local model.[2] RCI and RI can also be accommodated with this approach, using $p(y_i|\vec{x_i}, X_{\mathcal{N}_i}, Y_{\mathcal{N}_i}, \theta_M)$ or $p(y_i|\vec{x_i}, X_{\mathcal{N}_i}, \theta_M)$, respectively, as the local model, assuming that $X_{\mathcal{N}_i} = \{\vec{x_j} | v_j \in \mathcal{N}_i\}$.

In this article, we use the pseudolikelihood based on Equation 3, as with most recent work; Section 9 discusses alternatives based on the joint likelihood. For future equations, we usually omit θ_M and assume it is implicitly conditioned on, e.g., writing simply $P(y_i|\vec{x_i}, Y_{\mathcal{N}_i})$ for the local model, except when explicitly discussing parameter learning. Note also that Equation 3 maximizes only over the known nodes V^K; Section 5 considers how to use the unlabeled nodes (V^U) with SSL.

Collective Inference: Given a learned model specified by parameters θ_M, collective inference can be performed using a variety of algorithms including belief propagation, Gibbs sampling, VMF (variational mean field), and ICA (Iterative Classification Algorithm). We later describe and present results with the latter three, amongst others, but focus on ICA.

ICA [Besag 1986; Neville and Jensen 2000; Lu and Getoor 2003a] is a simple, popular, and effective algorithm [Sen et al. 2008; Bilgic et al. 2010; McDowell and Aha 2012], and is very similar algorithmically to VMF. We describe ICA's use with CI; later sections consider how to add neighbor attributes to produce RCI. ICA first predicts a label for every node in V^U using only self attributes, via a "bootstrap classifier" M_A. It then constructs additional "relational" (i.e., link-based) features using the known and predicted node labels (Y^K and Y^U). Next, ICA re-predicts labels for V^U using both self attributes and relational feature values, via the local conditional model, which we call M_{AL} (AL for *attributes* and *labels*). This process of feature computation and prediction is repeated, e.g., until convergence or for a fixed number of iterations.

Hybrid Models: In most prior work, the local conditional model M_{AL} is a single "unified" classifier that directly predicts a label y_i for some node v_i. For instance, with our running webpage example, M_{AL} could use a logistic regression classifier with features based on the frequency of certain words being found on page v_i (i.e., the self attributes, $\vec{x_i}$), as well as features that are aggregations such as "count the number of Students" for all of v_i's neighbors' labels ($Y_{\mathcal{N}_i}$). However, McDowell and Aha [2012] showed that higher accuracy can often be achieved via the use of a "hybrid" classifier that combines one classifier for $\vec{x_i}$ and another classifier for features based on $Y_{\mathcal{N}_i}$ (cf., the similar informal account given by Lu and Getoor [2003a]). In particular, if we make the common assumption that $\vec{x_i}$ and $Y_{\mathcal{N}_i}$ are conditionally independent given y_i, we can then compute the combined prediction

[2]This formulation is known as a *relational dependency network* (RDN) [Neville and Jensen 2007]. If we choose to use logistic regression as the local conditional model, then this is also a form of relational Markov network (RMN) [Taskar et al. 2002] where learning has been simplified via the use of pseudolikelihood, as is frequently done [Bilgic et al. 2010; Namata et al. 2011].

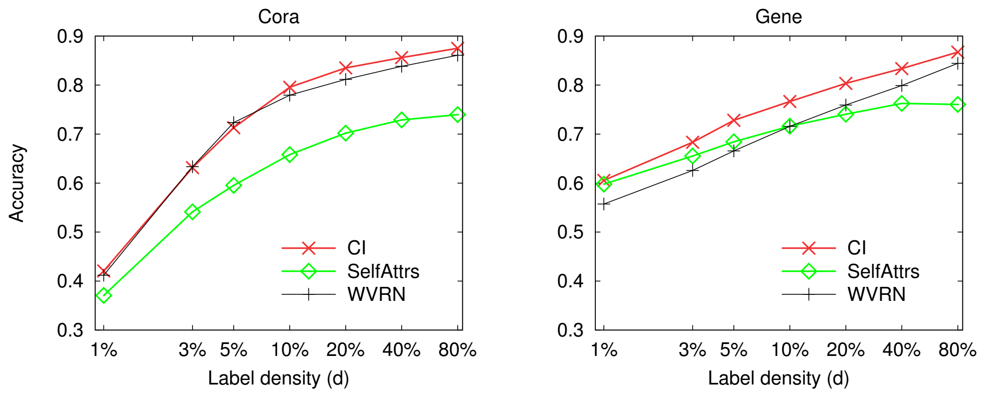

Fig. 1. Average classification accuracy for the "Cora" and "Gene" datasets, using logistic regression with ICA (for CI) and no SSL (see Section 5). The x-axis uses a log scale.

$$\begin{aligned} p(y_i|\vec{x_i}, Y_{\mathcal{N}_i}) &= \frac{p(\vec{x_i}|y_i)p(Y_{\mathcal{N}_i}|y_i)p(y_i)}{p(\vec{x_i}, Y_{\mathcal{N}_i})} \\ &= \frac{\frac{p(y_i|\vec{x_i})p(\vec{x_i})}{p(y_i)} \frac{p(y_i|Y_{\mathcal{N}_i})p(Y_{\mathcal{N}_i})}{p(y_i)} p(y_i)}{p(\vec{x_i}, Y_{\mathcal{N}_i})} \\ &= \alpha \frac{p(y_i|\vec{x_i})p(y_i|Y_{\mathcal{N}_i})}{p(y_i)} \end{aligned} \quad (4)$$

where α is a normalizing constant independent of y_i.

Using such a hybrid classifier enables us to choose a different type of classifier for the attributes vs. for the relational features, and also facilitates different regularization strategies for these different types of features. In addition, we later build upon this technique to enable the addition of neighbor attributes to the model. Based on the positive results of McDowell and Aha [2012], we use the hybrid classifier method throughout this article.

3. THE CHALLENGE OF LABEL SPARSITY

This section briefly highlights why label sparsity makes LBC so challenging, and outlines the methods we will consider for increasing its accuracy in the presence of such sparsity.

Of the LBC models shown in Table I (including CI, RI, and RCI), CI has been by far the most popular (we discuss why in the next section). Figure 1 plots the average accuracy of CI for Cora and Gene (two common LBC datasets, see Section 6) as the label density d (the fraction of nodes with known labels) varies. We compare CI with two baselines: wvRN, a common "neighbor labels only" baseline for LBC (see Section 2), and SelfAttrs. Here, CI and SelfAttrs use logistic regression, while wvRN does not require any learned model.

When the labels are very dense (i.e., $d = 80\%$), both LBC methods (CI and wvRN) perform well. Specifically, they leverage the dataset's link-based correlations to produce much higher accuracy than SelfAttrs, which simply ignores links. When the labels are very sparse, however, accuracy decreases substantially for all three methods. Notably, CI's accuracy (as well as wvRN's) decreases much more quickly than SelfAttrs's does, so that when $d = 1\%$ it provides only a small gain compared to SelfAttrs. Why does CI's accuracy decrease so sharply as d decreases, and more sharply than that of SelfAttrs? In addition, why does CI no longer provide any gain compared to wvRN on Cora, even though wvRN ignores attributes and performs no learning?

First, during CI's inference, many neighbor labels are unknown. Thus, a potential problem for CI is that some predicted labels used during inference will be incorrect, decreasing overall accuracy. However, prior work shows that CI—when learning uses a fully-labeled graph—can be effective even when *all* labels in a separate test graph are initially unknown [Neville and Jensen 2007; McDowell et al. 2009]. Thus, having a large number of unknown labels during CI's *inference*, while a limitation, is not enough to explain CI's substantial performance decline.

An additional, and substantial, problem is that, at low label density, CI struggles to *learn* the parameters related to "neighbor label"-based features. Links can be used for learning such features only where *both* nodes of a link have known labels (this enables the estimation of values such as "If v is labeled `Student`, what is the probability that a neighbor of v is labeled `Professor`?"). For example, when the label density is 10%, perhaps only 1% of links will connect two nodes that both have known labels. In this case, CI will be able to learn the parameters related to its label-based features using only about 1% of the links (unless some kind of SSL is used), and overall accuracy will suffer.

Naturally, accuracy decreases for any method when less information is available for learning, but the decrease in accuracy is especially acute for CI when the network is very sparsely labeled, due to these challenges with relational features. If this low accuracy is not adequate for a particular task, how can be it improved? We explore two primary approaches in this article. First, Section 4 explains how to leverage neighbor attributes as model features, first via existing techniques and then via a new technique that enables them to be used, for the first time, in conjunction with popular discriminative classifiers. Later results will show that, contrary to prior expectations and practice, such usage (in both forms) can markedly increase accuracy. Second, Section 5 explores SSL, specifically explaining how learning from predicted labels can improve learning and the resultant accuracy.[3]

Ultimately, we show that the low accuracy of Figure 1 is *not* an inevitable consequence of sparse labeling. For instance, Section 7 will demonstrate some accuracy gains of more than 30% for Cora, with additional significant increases for other datasets.

4. LEVERAGING NEIGHBOR ATTRIBUTES FOR LBC

This section first explores why neighbor attributes have been very rarely used for LBC. Next, we describe existing methods for using such attributes, then introduce a new method that allows them to be used for the first time with discriminative classifiers such as logistic regression.

4.1. Prior Work with Neighbor Attributes

Some early work on LBC evaluated models that included neighbor attributes [Chakrabarti et al. 1998; Taskar et al. 2001]. However, recent work has used such models (including RI and RCI) very rarely for two primary reasons. First, prior work found that, while using neighbor labels can increase accuracy, using neighbor attributes can actually *decrease* accuracy [Chakrabarti et al. 1998]. Later, Jensen et al. [2004] compared CI vs. RI and RCI. They found that CI somewhat outperformed RCI and performed much better than RI. They described how neighbor labels provide a "clever factoring of the space of dependencies," while neighbor

[3]Appendix B examines a third area, by considering whether more computationally expensive inference algorithms (such as Gibbs sampling or VMF) might increase accuracy.

Table III. Related work on LBC that used some variant of semi-supervised ICA or VMF (with sparsely-labeled networks). The first row is an exception; it used fully-labeled training networks but is included for comparison.

	Classifier	RCI	RI	CI	SelfAttrs	NeighAttrs	NeighLabels	Inference alg.	Datasets used
Jensen et al. [2004]	RBC	✓	✓	✓	✓			Gibbs	Gene, synthetic
Lu and Getoor [2003b]	LR			✓	✓			ICA	Citeseer, Cora
Xiang and Neville [2008]	Dec. tree	✓					✓	VMF	Gene, synthetic
Bilgic et al. [2010]	LR			✓	✓			ICA	Citeseer, Cora
Shi et al. [2011]	LR			✓				ICA	Citeseer, Cora, Gene
McDowell and Aha [2012]	LR			✓	✓		✓	ICA	Citeseer, Cora, Gene
Pfeiffer III et al. [2014b, 2015]	NB & LR			✓	✓		✓	Gibbs, VMF	DVD, Facebook, IMDB, Music
This article	NB & LR	✓	✓	✓	✓	✓	✓	ICA, Gibbs, VMF Cautious ICA	Citeseer, Cora, Gene, HepTH, IMDB, PubMed

attributes greatly increase the parameter space, leading to lower accuracy, especially for small training graphs. Note that the number of additional parameters with RI is directly proportional to the number of attributes, which is (potentially) a significant issue with, for instance, our running webpage example, where thousands of distinct words (attributes) might be relevant for prediction.

Second, it is unclear how to include neighbor attributes in many popular classifiers. In particular, nodes usually have a varying number of neighbors. Thus, with neighbor attributes as features, there is no direct way to represent a node in a fixed-sized feature vector as expected by classifiers such as logistic regression or support vector machines. With neighbor labels, CI algorithms address this issue with aggregation functions (such as "Count") that summarize all neighboring labels into a few feature values. This works well for labels, which are discrete and highly informative, but is more challenging for attributes, which are more numerous, may be continuous, and are individually less informative than a node's label. Thus, this approach fared very poorly in early work [Chakrabarti et al. 1998]; we consider it again in Section 8.3.

These two factors have produced a prevailing wisdom that CI based on neighbor labels is better than RI based on neighbor attributes (cf., Sen et al. 2008; Rossi et al. 2012). This conclusion rested on studies with fully-labeled training graphs, but has been carried into the important domain [Gallagher et al. 2008; Shi et al. 2011] of sparsely-labeled graphs. In particular, Table III summarizes the models used by the most relevant prior work with such graphs. Only one study [Xiang and Neville 2008] used models with neighbor attributes (e.g., with RI or RCI), and it did not evaluate whether they were helpful.[4]

Our results will later show that this prevailing wisdom was partly correct, but that, for sparsely-labeled networks, neighbor attributes are often much more useful than previously thought. Specifically, we will address both factors described above that seemed troublesome with neighbor attributes. First, we will show that when using a classifier that naturally accommodates a varying number of neighbors (such as naive Bayes), using neighbor attributes with RI or RCI often substantially increases accuracy compared to CI, *if* labels are sparse (Section 7.1). Second, we will demonstrate that our new method, Multi-Neighbor Attribute Classification (MNAC), which enables any probabilistic classifier to be used with neighbor attributes, often leads to further and significant accuracy gains (Section 7.2).

[4]Prior work with neighbor attributes, including that of Xiang and Neville [2008], used methods like decision trees or naive Bayes that do not require a fixed-length feature vector.

The next section explains prior techniques for using neighbor attributes with a classifier such as naive Bayes, then Section 4.3 describes our new MNAC method.

4.2. Existing Methods for Neighbor Attributes

Recall that $\mathcal{N}_i$ is the set of nodes adjacent to v_i. Furthermore, let $X_{\mathcal{N}_i}$ be the set of attribute vectors for all nodes in $\mathcal{N}_i$ ($X_{\mathcal{N}_i} = \{\vec{x_j} | v_j \in \mathcal{N}_i\}$).

Suppose we wish to predict the label y_i for v_i based on v_i's attributes and the *attributes* of $\mathcal{N}_i$ (for example, the words in page v_i, and the words in the pages that v_i links to). As described in Section 4.1, the variable size of $\mathcal{N}_i$ presents a challenge. To address this general issue, prior studies (with neighbor *labels*) often assume that the labels of nodes in $\mathcal{N}_i$ are conditionally independent given y_i. This assumption is not necessarily true, but often works well in practice [Jensen et al. 2004; Neville and Jensen 2007; McDowell et al. 2009]. In our context (with neighbor *attributes*), we can make the analogous assumption that the attribute vectors of the nodes in $\mathcal{N}_i$ (and the attribute vector $\vec{x_i}$ of v_i itself) are conditionally independent given y_i. Using Bayes rule and this assumption yields

$$\begin{aligned} p(y_i|\vec{x_i}, X_{\mathcal{N}_i}) &= p(y_i)\frac{p(\vec{x_i}, X_{\mathcal{N}_i}|y_i)}{p(\vec{x_i}, X_{\mathcal{N}_i})} \\ &= p(y_i)\frac{p(\vec{x_i}|y_i)}{p(\vec{x_i}, X_{\mathcal{N}_i})}\prod_{v_j \in \mathcal{N}_i} p(\vec{x_j}|y_i) \\ &= \alpha \cdot p(y_i)p(\vec{x_i}|y_i)\prod_{v_j \in \mathcal{N}_i} p(\vec{x_j}|y_i) \qquad (5) \end{aligned}$$

where α is a normalizing constant independent of y_i.

To use Equation 5, we must compute $p(\vec{x_i}|y_i)$ and $p(\vec{x_j}|y_i)$. The same technique works for both; we now explain for the latter. We further assume that all attribute values for v_j (e.g., the values inside $\vec{x_j}$) are independent given y_i. If nodes have N_A attributes, then

$$\begin{aligned} p(\vec{x_j}|y_i) &= p(x_{j1}, x_{j2}, ..., x_{jN_A}|y_i) \\ &= \prod_{k=1}^{N_A} p(x_{jk}|y_i) \end{aligned}$$

Plugging this equation (and the equivalent one for $p(\vec{x_i}|y_i)$) into Equation 5 yields a (relational) naive Bayes classifier [Neville et al. 2003a]:

$$p(y_i|\vec{x_i}, X_{\mathcal{N}_i}) = \alpha \cdot p(y_i)[\prod_{k=1}^{N_A} p(x_{ik}|y_i)][\prod_{v_j \in \mathcal{N}_i}\prod_{k=1}^{N_A} p(x_{jk}|y_i)].$$

In particular, the features used to predict the label for v_i are v_i's attributes and v_i's neighbors' attributes, and these values are assumed to be conditionally independent. Jensen et al. [2004] used this classifier for RI, and a simple extension to add the labels of neighboring nodes as features yields the equations needed for RCI.

4.3. Multi-neighbor Attribute Classification

The method described above can predict with neighbor attributes, and can increase classification accuracy, as we show later. However, it has two potential problems. First, it ignores dependencies among the attributes within a single node. Second, it requires using probabilities like $p(\vec{x_j}|y_i)$, whereas a discriminative classifier (e.g., logistic regression) would compute $p(y_i|\vec{x_j})$. Thus, the set of classifiers that can be used is constrained, and overall accuracy may suffer.

A new idea is to take Equation 5 and apply Bayes rule to each conditional probability separately. This yields

$$p(y_i|\vec{x_i}, X_{\mathcal{N}_i}) = \alpha \cdot p(y_i) \frac{p(y_i|\vec{x_i})p(\vec{x_i})}{p(y_i)} \prod_{v_j \in \mathcal{N}_i} \frac{p(y_i|\vec{x_j})p(\vec{x_j})}{p(y_i)}$$
$$= \alpha' \cdot p(y_i|\vec{x_i}) \prod_{v_j \in \mathcal{N}_i} \frac{p(y_i|\vec{x_j})}{p(y_i)} \quad (6)$$

where the last step folds all values independent of y_i into α'.

We refer to classification based on Equation 6 as *Multi-Neighbor Attribute Classification* (MNAC). This approach requires two conditional models, $p(y_i|\vec{x_i})$ and $p(y_i|\vec{x_j})$. The first is a standard (self) attribute-only classifier, while the second is more unusual: a classifier that predicts the label of a node based on the attributes of *one* of its neighbors. Because the prediction for this latter model is based on just a single attribute vector, any probabilistic classifier can be used, including discriminative classifiers such as logistic regression.

Learning and Inference with MNAC**:** For learning the second model, $p(y_i|\vec{x_j})$, a single training example is the known label y_i of node $v_i \in V^K$, combined with the attribute vector of some *other* node v_j where $v_j \in \mathcal{N}_i$.[5] In particular, the training data is $D = \{< \vec{x_j}, y_i > | \; v_i \in V^K \wedge v_j \in \mathcal{N}_i\}$. Thus, if a known node $v_i \in V^K$ links to five other nodes, it will generate five training examples (independent of whether those five nodes have known labels or not). To prevent very high-degree nodes from disproportionately impacting the learning, we weight each example by the reciprocal of $degree(v_i)$ (e.g., those five examples would each have weight $\frac{1}{5}$, so that their total weight sums to one); Section 8.3 considers alternatives. Finally, if node v_j links to two known nodes $v_{i1}, v_{i2} \in V^K$ and $y_{i1} = y_{i2}$, then the two examples generated ($< \vec{x_j}, y_{i1} >, < \vec{x_j}, y_{i2} >$) will be identical and can be combined into one example with a larger weight. Using this observation, we can reduce the worst case number of distinct examples from $O(|E|)$ to $O(|V| \cdot N_C)$, where N_C is the number of class labels ($2 \leq N_C \leq 7$ for our datasets).

During inference, to predict a label for each v_i, Equation 6 will evaluate $p(y_i|\vec{x_j})$ once for each of v_i's neighbors (a total of $O(|E|)$ evaluations). However, for a fixed node v_j, $p(y_i|\vec{x_j})$ is always the same, so simple caching reduces the total number of evaluations needed to $O(|V|)$, once for every node $v_j \in V$. For RI, these results are then combined with $p(y_i|\vec{x_i})$ to produce an overall prediction $p(y_i|\vec{x_i}, X_{\mathcal{N}_i})$ using Equation 6. As with the relational naive Bayes classifier discussed in Section 4.2, adding features based on the *labels* of neighboring nodes to Equation 6 (e.g., with "multiset" features, see Section 6.2) instead yields predictions based on a RCI model.

Discussion: The derivation for MNAC is simple, but it has not been used for any prior work on LBC (excluding our own preliminary study, McDowell and Aha 2013). The closest related work is that of McDowell and Aha [2012], who used a similar technique to produce "hybrid models" that combine two classifiers (one based on self attributes and one on neighbor labels), as described in Section 2.1. The derivation is different, however, and that work does not consider neighbor attributes.

The derivations above used one or two independence assumptions that may *not* hold in many cases. Do they nonetheless yield effective methods for leveraging neighbor attributes on real, sparsely-labeled datasets? In addition, it is not obvious that MNAC will increase accuracy vs. the existing method based on naive Bayes. For instance, McDowell and Aha [2012] found that naive Bayes performs better than logistic regression

[5]In this section, we are assuming that no SSL is used, and thus need $v_i \in V^K$. In experiments later that do use SSL, this restriction is not necessary, i.e., $v_i \in V$, and there are many more training examples.

for leveraging neighbor *labels*; which would be best for neighbor *attributes*? Section 7 answers these questions.

5. INFERENCE AND SEMI-SUPERVISED LEARNING FOR LBC

This section briefly summarizes necessary information on inference, then considers how to use the unlabeled portion of the graph to improve learning for CI, RI, or RCI.

5.1. Collective Inference for LBC

Section 4 described how to predict a node's label based on self attributes and neighbor attributes, i.e., with RI. We assume that all attributes have known values, so RI's inference is trivial. For instance, given logistic regression models for $p(y_i|\vec{x_i})$ and $p(y_i|\vec{x_j})$, Equation 6 can simply be evaluated once for each node, yielding the final result.

If, however, neighbor labels are being used (with RCI or CI), then some form of collective inference will be needed, since many such labels will not have known values. In the body of this article, we use the popular ICA algorithm (see Section 2.1). We also evaluated several other inference algorithms, including Gibbs sampling and variational mean field (VMF). We found, however, that none of these methods consistently outperformed ICA. To streamline the article, we therefore defer the description and evaluation of different inference algorithms, as well as discussion of the computational complexity of these methods and CI vs. RI vs. RCI, to Appendix B.

5.2. Learning with Expectation Maximization and Related Variants

Many LBC variants use local conditional models like M_A and M_{AL} that were described in Section 2.1 and are used for inference by ICA and other methods. Most prior LBC approaches assume that M_A (which uses only self *attributes*) and M_{AL} (which also uses *linked* neighbors) are learned from a separate, fully-labeled training graph. For our within-network task, however, we assume that there is a single sparsely-labeled graph. In this case, learning these classifiers is challenging because of label sparsity. Learning M_{AL} is especially problematic when it includes features based on neighbor labels (e.g., with CI or RCI), since relational (label-based) features can only be used for learning in the rare case where *both* node endpoints of a link have known labels (see Section 3). If neighbor attributes are used instead (with RI), then sparsity is less of a crippling problem (see comparison in Section 7), but label sparsity still greatly reduces the number of nodes that can be directly used for learning, limiting accuracy.

Given a large set of nodes but only a small set of provided labels, it is natural to consider some sort of semi-supervised learning (SSL), where predicted labels are used to augment the learning process. A few researchers have investigated this scenario, and Table III summarizes the most relevant studies, which all use some variant of semi-supervised ICA (or the related VMF). In particular, some of the work referenced in Table III has used a form of Expectation Maximization (EM) [Dempster et al. 1977]. In our context, EM repeatedly predicts the expected values of the unknown labels, then updates the model parameters θ_M while using the expected values of the unknown labels. For CI, these two steps are (expanding upon Equations 1 & 3):

E-Step: evaluate $P(Y^U|V, E, X, Y^K, \theta_M^{k-1})$

M-Step: $\theta_M^k = \arg\max_\theta P(Y^U|V, E, X, Y^K, \theta_M^{k-1}) \sum_{v_i \in V^K} \log P(y_i|\vec{x_i}, Y_{\mathcal{N}_i}, \theta_M^{k-1})$

While some of the work cited in Table III has used EM directly, we observe that *all* of these works used a variant of SSL that can be described by a single learning algorithm that generalizes EM. Figure 2 shows this algorithm; we now explain the learning variants of Table III in terms of this algorithm. Note that this algorithm also

```
SSL_learn ($V, E, X, Y^K, N_{Lrn}, LearnType$)=
// $V$=nodes, $E$=edges, $X$=attribute vectors, $Y^K$=labels of known nodes ($Y^K = \{y_i | v_i \in V^K\}$)
// $N_{Lrn}$=# of learning iterations, $LearnType$=HARD or SOFT
1   $M_A \leftarrow learn(X^K, Y^K [, V, E, X])$            // w/ known labels, learn self (+possibly neigh.) attrs. model
2   $Y^U \leftarrow predict(X^U, M_A\ [, V, E, X])$          // Predict with just attributes; result $Y^U = \{\vec{p_i} | v_i \in V^U\}$
3   for $k$ = 1 to $N_{Lrn}$ do
4       if ($LearnType$ = SOFT)                             // Select node labels to use below: the known labels plus...
            $Y' \leftarrow Y^K \cup \{\vec{p_i} | v_i \in V^U\}$           //    label probabilities for each node in $V^U$
        else $Y' \leftarrow Y^K \cup \{argmax\ \vec{p_i} | v_i \in V^U\}$  //    OR the most likely label for each node in $V^U$
5       for each node $v_i \in V$ do                        // Compute label-based feature values for each $v_i$, using
            $\vec{f_i} \leftarrow calcLabelFeats(V, E, Y', i)$           //    labels and/or label probabilities selected above
6       // Using all labels, (re)learn the two models based on...
        $M_{AL} \leftarrow learn(X, Y'\ [, V, E], \{\vec{f_i} | v_i \in V\})$  // ...attrs. (self and possibly neighbor) and label features
        $M_A \leftarrow learn(X, Y'\ [, V, E])$                // ...attrs. (self and possibly neighbor) only
7       // Run single-step or collective inference; obtain predictions $Y^U$ where $Y^U = \{\vec{p_i} | v_i \in V^U\}$
        $Y^U \leftarrow predict(V, E, X, Y^K, M_{AL}, M_A$, <inference options>)
8   return $Y^U$
```

Fig. 2. Generic SSL learning for LBC. Variables with superscripts of "K" relate to "known nodes"; "U" superscripts relate to nodes with unknown labels. For RI, step 5 (and half of step 6) is not needed, since RI does not use any label-based relational features. Brackets such as $[, V, E]$ indicate optional arguments that are needed only when neighbor attributes are used (with RI or RCI).

expands upon the E and M steps shown above to illustrate how RCI and RI, not just CI, can be used.

The SSL variants of Table III first learn an attribute-only classifier M_A from the known labels, then predict labels for the unknown nodes V^U with M_A (Steps 1-2 in Figure 2).[6] Step 4 then collects the known and predicted labels into Y', which Step 5 uses to compute relational feature values. Next, Step 6 uses all known and predicted labels, attributes, and relational feature values to (re)learn the classifiers M_A and M_{AL}. Step 7 then uses M_A and M_{AL} to predict new labels (for RCI or CI, this will involve collective inference with, for example, ICA, VMF, or *Gibbs*). Steps 4-7 may possibly be repeated N_{Lrn} times, before Step 8 returns the final set of label predictions.

When Steps 4-7 are executed only once ($N_{Lrn} = 1$), we call this method SSL-ONCE; this was used by Shi et al. [2011] and Bilgic et al. [2010]. Alternatively, more complex variants perform a form of EM where the algorithm repeatedly estimates new labels given the current models (i.e., Steps 2 & 7 are the E-step) and then maximizes the probability of a new model given the current label estimates (i.e., Steps 4-6 are the M-step). We call this approach SSL-EM. Xiang and Neville [2008] and Pfeiffer III et al. [2014b, 2015] use this method with a fixed number of iterations (N_{Lrn} = 10), while Lu and Getoor [2003b] repeat based on a convergence condition.[7]

As described above, prior work has generally used either one or many learning iterations. Perhaps surprisingly, we later show that an interesting additional choice is SSL-TWICE, where $N_{Lrn} = 2$. For RI and RCI, this usually has only a small impact compared to $N_{Lrn} = 1$, but for CI the additional learning iteration sometimes has a substantial positive impact. Observe that, with CI, label prediction in Step 2 involves only self-attributes, and thus for the first iteration, the model learning in Step 6 uses

[6]In all prior work (except that of Xiang and Neville 2008), this step involved *only* self attributes, but if RI or RCI is used then neighbor attributes will also be used here. Neighbor labels are not helpful at this point because typically very few are known.

[7]Bilgic et al. [2010], Xiang and Neville [2008], and Pfeiffer III et al. [2014b] actually perform Step 6 differently, using only the nodes in V^K, not V^U, for the learning in Step 6. McDowell and Aha [2012] and Pfeiffer III et al. [2015] show that Figure 2's method (using all available nodes) generally performs best (if appropriate corrections for possible label imbalances are applied), and so we adopt it here (cf., Section 8.1).

Table IV. Data sets summary.

Characteristics	**Cora**	**CiteSeer**	**Gene**	**HepTH**	**IMDB**	**PubMed**
Total nodes	2708	3312	1103	2194	5140	19727
Total links	5278	4536	1672	9752	68084	44324
Average node degree	3.9	2.7	3.0	8.9	26.5	4.5
Label consistency	81%	74%	83%	67%	43%	80%
Label autocorrelation	0.88	0.84	0.81	0.53	0.18	0.83
# class labels	7	6	2	7	3	3
% dominant class	16%	21%	56%	36%	44%	40%

label predictions that ignored all links. These label predictions are very helpful for CI (since they enable all links to be used for learning), but they have not yet had the opportunity to potentially benefit from any link-based correlations. Such link-based correlations *are* used when collective inference is run in Step 7 (hopefully improving accuracy)—and thus first affect learning during step 6 of the *second* iteration of the loop in Figure 2. Thus, having at least two learning iterations can be especially helpful for CI. We therefore include results with SSL-TWICE (and SSL-EM) to be fair to CI compared to RI and RCI (see Section 7.3).

Finally, most SSL approaches for LBC have used a "hard" labeling approach: Steps 2 and 7 predict the single most likely label for each node, which is then used for feature computation (Step 5) and learning (Step 6). In contrast, some studies (Xiang and Neville [2008]; Pfeiffer III et al. [2014b, 2015]) maintain "soft" probability estimates of the predicted labels for each node in V^U, and use these for learning (though they did not evaluate whether the "soft" learning was helpful). We thus evaluate additional "SOFTLRN" variants, which is the most consistent with the typical formulations of Expectation Maximization. Specifically, with these learning variants, Step 4 maintains the soft probability estimates for each node in V^U, these estimates are used for (weighted) feature computation in Step 5, and Step 6 uses these weighted features, along with appropriate weights for each node in V^U based on its estimated label probabilities, to influence learning.

Section 7 compares these variants to discover whether the additional complexity of multiple EM-like iterations and/or soft learning improves overall accuracy. Section 9 discusses alternative methods that are not based on the general approach of Figure 2.

6. EXPERIMENTAL METHOD

6.1. Datasets and Features

While a wide variety of link-based datasets exist, acquiring such *labeled* datasets, as needed for evaluation, is more challenging (see Section 1). Table IV shows the six real datasets that we consider, which includes the datasets most commonly used in prior work (cf., Table III) as well as some others. We removed all nodes with no links, but we did not (as some others did) use only the largest connected component of the graphs.

Cora (cf., Sen et al. 2008) is a collection of machine learning papers and **Citeseer** (cf., Sen et al. 2008) is a collection of research papers; the task is to predict the topic (class label) of each paper. Attributes represent the presence of certain words, and links indicate citations. We mimic Bilgic et al. [2010] by ignoring link direction, and also by using the 100 top attribute features after applying PCA to all nodes' attributes.

Gene (cf., Jensen et al. 2004) describes the yeast genome at the protein level; links represent protein interactions. We mimic Xiang and Neville [2008] and predict protein localization using four attributes: Phenotype, Class, Essential, and Chromosome. With logistic regression we binarized these, yielding 54 attributes.

HepTH is a set of journal articles on high-energy physics, as processed by McDowell et al. [2009]; links represent citations. The task is to predict the topic of each article.

Attributes represent the presence of words in the article title or name of the corresponding journal; PCA was again used to produce the top 100 attribute features.

IMDB is a dataset drawn from the Internet Movie Database (www.imdb.com), where each node is a movie, as created by Kuwadekar and Neville [2011]. They linked movies that had the same producer and considered only the years 2001–2007; we link movies that have the same studio (cf., Neville and Jensen 2005) and consider all movies in the dataset (years 1980–2007). The task is to predict the (inflation-adjusted) box-office earnings for each movie as either a blockbuster (earnings $>$ \$60 million), flop (earnings $<$ \$10 million), or other. This is a challenging prediction task with few useful attributes; we use attributes based on the movies' genre (using the top 8 values including comedy, action, drama, etc.) and also the number of movies made by the movie's director. Because studios change over time, we ignore links between movies whose release year differed by more than one.

PubMed (cf., Namata et al. 2012) is a collection of medical research papers regarding one of three types of diabetes (thus, there are three possible class labels). Links represent citations. The original attributes represent the frequency of the most common 500 words, which were transformed by PCA to produce the top 100 attribute features.

Table IV also contains relevant statistics about each of the datasets. Label autocorrelation is a measure of the correlation between the class labels of linked nodes (specifically, using Pearson's corrected contingency coefficient, cf., Jensen and Neville 2002). Label consistency is the fraction of links that connect two nodes with the same class label; this measures the amount of *homophily*, the most common (though not the only) form of correlation between linked nodes. LBC is most useful when significant autocorrelation is present, so we focus on datasets with higher autocorrelation values (as with prior studies), though also include one dataset with low but non-zero correlation (IMDB) and one dataset with moderate autocorrelation (HepTH).

We focus on cases where the attributes are at least moderately predictive. Thus, we did not use previous datasets, e.g., based on Flickr and BlogCatalog [Tang and Liu 2009], where this is not true; future work should study this other case more closely.

6.2. Classifiers and Regularization

We evaluate the six models shown in Table I, focusing on the first three. All except NEIGHLABELS require learning a classifier to predict the label based on self attributes and/or a classifier to predict based on neighbor attributes. For these classifiers, we evaluate naive Bayes, because of its past use with neighbor attributes [Jensen et al. 2004], and logistic regression, because it usually outperformed naive Bayes [Sen et al. 2008; Bilgic et al. 2010]. For neighbor attributes, logistic regression uses the new MNAC method.

RCI and CI also require a classifier to predict based on neighbor labels. McDowell and Aha [2012] found that naive Bayes with "multiset" features was superior to logistic regression with "proportion" features as used by Bilgic et al. [2010]. Thus, we use naive Bayes for neighbor labels, and combine these results with the naive Bayes or logistic regression classifiers used for attributes (described above), using the "hybrid model" method described in Section 2.1.

Later sections mention the number of parameters used by each classifier. Assume there are N_A attributes and N_C classes. A logistic regression classifier requires about $N_A N_C$ parameters to predict the class label based on self attributes, or the same number of parameters to predict based on neighbor attributes. To make predictions based on neighbor labels, a naive Bayes classifier using the "multiset" approach described above needs about $(N_C)^2$ parameters (roughly counting how many times in the training data a node with label i links to a node with label j, for $i, j \in C$). Thus, the hybrid model described above requires about $N_A N_C + (N_C)^2$ parameters for CI, $2N_A N_C$ parameters for RI, and $2N_A N_C + (N_C)^2$ parameters for RCI. For example, if $N_A = 100$ and $N_C = 5$, then these are 525, 1000, and 1025 parameters, respectively.

For sparsely-labeled data, the choice of regularization parameters can have a large impact on accuracy. To ensure fair comparisons, we used five-fold cross-validation on the labeled data, selecting the value of the regularization hyperparameter that maximized accuracy on the held-out labeled data.[8] In particular, we used a Gaussian prior for all logistic regression parameters, with some variance σ^2 chosen via cross-validation. For naive Bayes, we used a Dirichlet prior on each discrete feature (with some hyperparameter α, as with McDowell et al. 2009), and a Normal-Gamma prior on each continuous feature (with zero mean, variance estimated from the observed data, and chosen hyperparameter τ).

6.3. Learning and Collective Inference

Unless otherwise specified, we use ICA (with 10 iterations) for inference (needed only for CI and RCI) and SSL-TWICE for semi-supervised learning, based on prior work's use of ICA and based on our results (later) demonstrating that these are consistently strong choices across the datasets. Section 7.3 and Appendix B considers the alternate forms of learning and inference, respectively. For SSL-EM, we mimic prior work [Xiang and Neville 2008; Pfeiffer III et al. 2014b, 2015] by using $N_{Lrn} = 10$ learning iterations. When using logistic regression, we also use "label regularization" [McDowell and Aha 2012] which biases the learning towards models that yield sensible label distributions (it does not apply to naive Bayes).

The three baseline models (SELFATTRS, NEIGHATTRS, NEIGHLABELS; see Table I) do not use SSL or label regularization. For NEIGHLABELS, we use WVRN, a common baseline based on relaxation labeling [Macskassy and Provost 2007], as described in Section 3.

6.4. Evaluation Procedure

We report accuracy averaged over 20 trials. For each trial, we randomly select some fraction of V (the "label density" d) to be "known" nodes V^K. The remaining nodes V^U have unknown labels and form the test set part of graph G. Section 7.2 also explores non-random label samplings. All algorithms are presented with the same 20 known/unknown splits. The attributes and links of "unknown" (test) nodes, but not their labels, are available during both learning and inference.

We focus on the important sparsely-labeled case [Gallagher et al. 2008; Shi et al. 2011; McDowell and Aha 2012], e.g., $d \leq 10\%$. To assess results, we use paired t-tests with a 5% significance level; below, "significant" or "significantly" always refers to this statistical test. For each value of d, however, the 20 test sets are not disjoint, and thus a traditional t-test may yield false conclusions. To compensate, we use the method of Wang et al. [2011], which reduces false positives to the expected level. This makes our results more conservative vs. uncorrected t-tests.

To provide some context for the amount of variability in each reported accuracy (variability over the different set of "known" nodes for each trial), Figures 3-5 include error bars. In particular, the total length of each error bar is twice the "standard error of the mean," which is appropriate since each plotted value represents the mean of 20

[8]The labeled data is divided into five folds. For each validation trial, we use four of the labeled folds as V^K, while all other graph nodes form V^U. Learning uses the labels from V^K, plus (via SSL) estimated labels from V^U. Inference accuracy, for cross-validation purposes, is evaluated using only the nodes in the fifth (held-out) fold. Validation accuracy is averaged over five runs, using each fold as the held-out data once.

trials.[9] As discussed above, however, the 20 trials are not entirely independent (since the test sets are not disjoint), and therefore this computation may underestimate the true standard error. For this reason, in the next sections we evaluate significance using the network-corrected, paired t-tests described above, rather than using unmodified t-tests or confidence intervals. Nonetheless, the standard errors shown in Figures 3-5 do give a general indication of variability. As we might expect, there is more variability when d is low and/or when "snowball sampling" (see Section 7.2.2) is used, while when d is high the combination of a substantial number of known labels, and averaging over 20 trials, results in much smaller variance for the estimated mean accuracies.

7. RESULTS

Sections 7.1 & 7.2 study the impact of different model choices (CI vs. RI vs. RCI), using default learning and inference settings (SSL-TWICE with ICA) that worked well across all of the datasets. Section 7.3 then examines the impact of different learning methods.

7.1. Using Neighbor Attributes with naive Bayes-based classifiers

Figure 3 plots the average accuracy of RCI, RI, and CI, using naive Bayes for the local attribute classifier, for four of the real datasets: Cora, Gene, HepTH, and PubMed (some results for Citeseer and IMDB are presented later; Citeseer has behavior very similar to Cora). The x-axis varies the label density d, and symbols encode statistical significance (see caption).

Result: With naive Bayes, RCI **and** RI **yield lower accuracy than** CI **when the labels are dense, but often significantly *higher* accuracy when the labels are sparse**. When label density is moderate or high ($\geq$ 20%), CI significantly outperforms both RI and RCI. High label density is similar to learning with a fully-labeled graph, and these results are consistent with those of Jensen et al. [2004]. However, when density is low or very low, CI is significantly worse than RI and RCI (with Cora and Gene), and is no better than RI and RCI on HepTH. [10] This is especially true for Gene – the same dataset used by Jensen et al. – where CI significantly trails RI and RCI for all $d < 10\%$. To the best of our knowledge, no prior work has reported this effect. Why does it occur?

To explain, we compare RI vs. CI in regards to CI's challenges with learning and inference as introduced in Section 3. First, for small d, CI's collective inference struggles with having to use predicted neighbor labels, which may often contain errors. In contrast, RI uses neighbor attributes, whose values are always known[11], and is also able to use exact (non-collective) inference. Second, CI's learning struggles with the parameters related to making predictions from neighbor labels. As explained in Section 3, learning such parameters requires links where *both* ends of the link have a (possibly predicted) label.[12] In contrast, with RI a single node's label makes a link useful for learning, since the node's neighbors' attributes are all known. Consider, for example,

[9]The standard error can be estimated as $SE = \frac{s}{\sqrt{n}}$, where n is the number of observations (in this case, 20 trials) and s is the standard deviation of the n accuracies that were averaged to obtain a particular plotted point.
[10]In Figure 3, for low d CI continues to produce the best accuracy with PubMed. Accuracy with PubMed is fairly high even when using only self attributes; this ameliorates CI's challenges with learning from sparse labels. Nonetheless, Section 7.2 shows that even for PubMed the best overall accuracy when d is low is often obtained by using RCI or RI, in conjunction with a logistic regression classifier instead of naive Bayes.
[11]We assume, like almost all related work, no missing attributes values (reasonable for webpages, documents, etc.); future work should consider other settings.
[12]Section 6.2 describes parameter computation for these neighbor label-based features more completely.

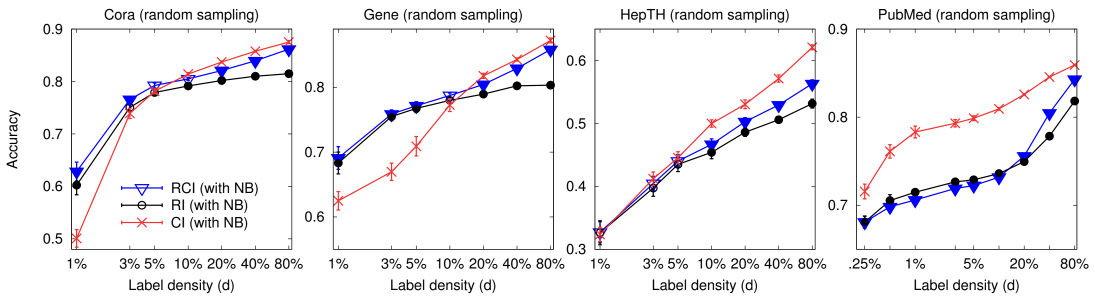

Fig. 3. Average accuracy using attribute classifiers based on naive Bayes (NB), using SSL-TWICE and *ICA*. Within a column, filled (non-hollow) circles or triangles indicate a significant difference (better or worse) vs. CI. Error bars are also included; see Section 6.4.

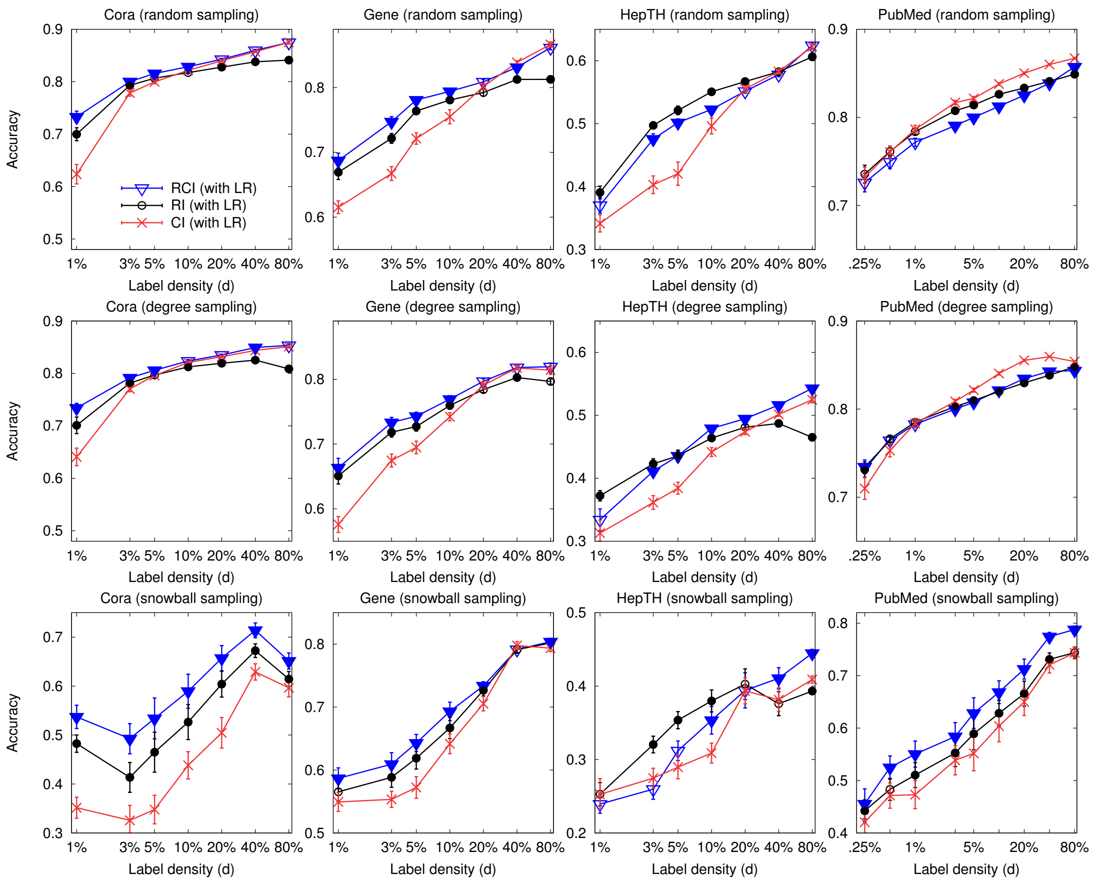

Fig. 4. Average accuracy using attribute classifiers based on logistic regression (LR), using SSL-TWICE and *ICA*. Different rows show results with random sampling, degree sampling, or snowball sampling. Since LR is used, RCI and RI use MNAC to incorporate neighbor attributes. Note the last row uses a different y-axis scale to show the full range of results. Filled symbols indicate a significant difference vs. CI. Error bars are also included; see Section 6.4.

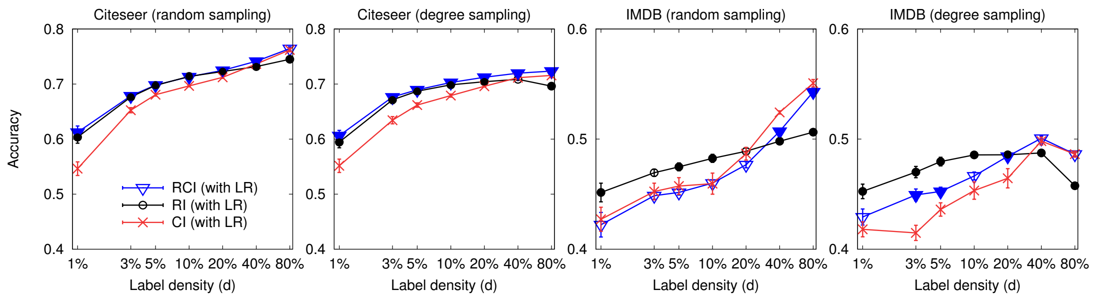

Fig. 5. Average accuracy using classifiers based on logistic regression (LR) for the Citeseer and IMDB datasets, using the same conditions as for Figure 4. Filled symbols indicate significant differences vs. CI. Error bars are also included; see Section 6.4.

Gene when $d = 10\%$ (110 known nodes). If (unlike in Figure 3) no SSL was being used, then CI could learn from only about 25 links, while RI could only learn from about 340 links[13]; for this reason RI's advantage compared to CI grows even larger when no SSL is used (see Section 7.3). If, as in Figure 3, effective SSL is used, then CI and RI can both learn from all the links, but CI has far fewer relational examples where the needed values (for labels and/or attributes) are all certain to be correct (e.g., 25 vs. 340 for the Gene example). More frequent errors among the training examples hamper learning, leading to reduced accuracy.

Thus, in Figure 3, RI often outperforms CI when d is low because of advantages in both learning and inference (and later improvements will extend these gains to additional datasets). Section 8 further analyzes RI (and RCI) vs. CI, and uses some synthetic data to analyze how the differences depend upon learning vs. inference issues.

7.2. Applying MNAC to Use Neighbor Attributes with Discriminative Classifiers

The previous section showed, for the first time, that using neighbor attributes (with RCI or RI) could significantly increase accuracy for some datasets. However, those results all used naive Bayes-based classifiers, whereas most recent work on LBC (see Table III) has instead used a discriminative classifier like logistic regression, and performance comparisons have generally found logistic regression to outperform naive Bayes for this domain [Sen et al. 2008; Bilgic et al. 2010]. Can neighbor attributes also increase LBC accuracy when logistic regression is used, now that Section 4 has demonstrated how to combine neighbor attributes with logistic regression via the new MNAC method?

7.2.1. Results with Random Sampling. Figure 4 shows the results when logistic regression is used for the attributes. The top row uses random sampling for the known nodes (as with Figure 3); we now discuss these results, and consider the next two rows in Section 7.2.2. Figure 5 also shows some results for Citeseer and IMDB.

Result: When using logistic regression instead of naive Bayes, overall accuracy generally increases, and the gains of RCI and RI vs. CI for sparse networks become even more pronounced. In general, Figures 4 & 5 (with logistic regression) display similar performance trends to Figure 3 (with naive Bayes): CI is significantly better than RI when the density is high, while RI performs relatively

[13]These values (25 and 340) were computed by averaging over the 20 trials, with random label sampling.

much better when the density is low. In Figures 4 & 5, however, the accuracy of RI (and usually RCI) with logistic regression improves in two ways.

First, RI now almost always improves over CI when d is low. Specifically, RI is significantly better than CI when $d \leq 10\%$ for Citeseer, Gene, HepTH, and (with one exception) IMDB, and when $d \leq 3\%$ for Cora. The gains are often substantial, e.g., for $d < 10\%$, RI's gain vs. CI averages 5.0% for Gene and 8.1% for HepTH.

Second, for RCI and RI using logistic regression with MNAC usually increases accuracy compared to using naive Bayes (compare the top row of Figure 4 to Figure 3). For instance, with Cora RCI is the best method with both logistic regression and naive Bayes when d is low, and using logistic regression with MNAC improves over naive Bayes by 2.3-10.4% (average of 5.4%) for $d < 10\%$. In general, logistic regression with MNAC increases the accuracy of RCI and RI substantially for Cora, Citeseer (results not shown), HepTH, and PubMed. For these datasets, logistic regression outperforms naive Bayes because the attributes are continuous (a challenging scenario for naive Bayes) and because logistic regression does not assume that the (many) attributes are conditionally independent. Thus, the new MNAC enables the use of a classifier better suited to the data, leading to higher accuracy than with naive Bayes. For Gene and IMDB, however, naive Bayes is generally better, likely because naive Bayes is able to use a smaller number of attributes to represent the same information. For instance, with Gene, naive Bayes can use the 4 discrete (but non-binary) attributes, whereas logistic regression must use the 54 binarized attributes.

Thus, whereas the results with naive Bayes showed that using neighbor attributes could sometimes increase LBC accuracy, using logistic regression with MNAC leads to more consistent gains for RCI/RI vs. CI and to (usually) higher overall accuracy. Naturally, the best classifier depends upon data characteristics; MNAC greatly expands the set of possibilities.

We use accuracy as our default performance metric, due to its simple interpretability and because the necessary corrections exist for paired significance tests based upon it (see Section 6.4). To assess whether our findings might be influenced by the choice of a different metric, Figure 6 presents results that are analogous to the top row of Figure 4, except that performance is measured by "multiclass AUC" (MAUC) [Hand and Till 2001], rather than accuracy. MAUC is independent of any assumptions about the relative mis-prediction costs for each class. The trends in Figure 6 are quite similar to those previously observed. One notable difference is that the relative performance of RI improves somewhat vs. both RCI and CI. Specifically, RI's MAUC often matches or exceeds that of RCI for some cases where RI's *accuracy* was lower (with Cora and Gene). Also, for low d, RI's MAUC is significantly higher than CI's on PubMed, whereas their accuracy was comparable. Figures 10-11 in Appendix B provide additional results with MAUC, and show similar trends.

Regardless of which metric is used, for sparsely-labeled networks, neighbor attributes appear to be much more useful than previously recognized, and our new logistic regression+MNAC often significantly increases accuracy (and MAUC). For simplicity, we consider only logistic regression below; Appendix B presents additional results with naive Bayes.

7.2.2. Results with other sampling methods. We assumed above that known labels are randomly sampled from the network—as with most work on LBC—but now consider two different patterns inspired by Xiang and Neville [2008]. First, "degree sampling" selects each node for labeling with probability proportional to its degree. Second, "snowball sampling" selects a single seed node for labeling, then uses a breadth-first search to select additional nodes. These variants may imitate some real-world patterns, since high-degree (prominent) nodes and/or certain subcommunities may be more likely

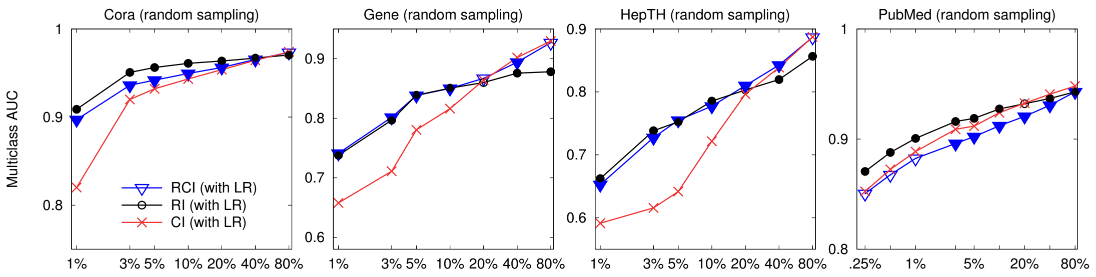

Fig. 6. Multiclass AUC using logistic regression, SSL-Twice, and *ICA*. These results mirror Figure 4, but with Multiclass AUC rather than accuracy. Filled symbols indicate a significant difference vs. CI, but the t-test corrections of Wang et al. [2011] do not apply with Multiclass AUC.

than others to be labeled. Results are shown in the last two rows of Figure 4, with some additional results for degree sampling in Figure 5. The sampling matters most when d is small, and we focus on this case. Three primary trends are described below.

Result: Degree sampling usually yields slightly lower accuray than random sampling. The decrease is substantial with HepTH and with CI on IMDB and PubMed (at least when d is small). Cora and Citeseer are the least affected; compared to HepTH, IMDB, and PubMed they have lower link density (and thus are less affected by the degree-based sampling).

Result: Compared to random sampling, snowball sampling leads to substantially lower accuracy for all datasets. In general, snowball sampling's breadth-first search will identify a particular region or cluster of the network for labeling. This sampling may negatively impact learning, since the sampled nodes may not be representative of the whole network, as well as inference, since the label clustering implies that many unlabeled nodes will have *no* links to known labels that can help to ground the inference of RCI and CI.

Result: When the known labels are non-randomly distributed, using RCI **or** RI **(instead of** CI**) is even more important for obtaining maximal accuracy.** For instance, for the one dataset (PubMed) where using neighbor attributes was not clearly helpful when random sampling was used, RCI and/or RI provide higher accuracy than CI if degree or snowball sampling is used, when d is small or very small. These gains are even more pronounced when measuring performance with MAUC (see Figure 11 in Appendix B). RCI or RI also become even more useful in a number of other cases, including Cora with snowball sampling and IMDB with degree sampling. Overall, of the 18 different scenarios considered by Figure 4 and Figure 5 (2 not shown), RCI and/or RI provide some significant accuracy gains vs. CI when d is small for all but one scenario.

Thus, using neighbor attributes in some form usually leads to higher accuracy than CI for most of our datasets when d is small, and this may be even more important if the known labels are not randomly distributed. In the remainder of this article we focus on the random sampling case; future work should continue to explore the impact of other scenarios.

7.3. The Impact of Semi-Supervised Learning for LBC

The results above all used SSL-Twice for learning and, where needed, ICA for collective inference. Table V examines results where we vary the learning algorithm. SSL-Once, SSL-Twice, and SSL-EM are all variants of the SSL algorithm of Figure

Table V. **Effect of learning:** Average accuracy for different combinations of models and learning algorithms, using logistic regression with ICA. Within each column, we bold the best result *and* all values that were *not* significantly different from that result. Label densities range from 1% to 10%, except for PubMed, which considers smaller densities because of its larger number of nodes.

	Citeseer				Cora				Gene			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	1%	3%	5%	10%
RCI+No-SSL	51.3	63.6	67.3	**71.0**	58.0	74.5	78.9	**82.8**	63.4	70.3	75.6	78.3
RCI+SSL-Once	59.1	67.2	**69.7**	**71.4**	71.3	**80.0**	**81.6**	**83.2**	67.1	73.6	**77.6**	**79.4**
RCI+SSL-Twice	61.2	**67.8**	**69.8**	**71.3**	**73.3**	**80.0**	**81.6**	82.9	68.7	**74.7**	**78.1**	**79.4**
RCI+SSL-Twice-SoftLrn	61.4	**68.1**	**69.6**	**71.4**	**71.9**	**80.0**	81.2	82.7	68.4	**75.3**	**77.1**	**79.7**
RCI+SSL-EM	**63.1**	**68.2**	**69.8**	**71.2**	**73.5**	**79.6**	**81.3**	82.7	**70.6**	**75.6**	**78.0**	**78.9**
RI+No-SSL	51.2	63.1	66.8	70.5	57.4	73.1	77.1	80.6	63.1	68.8	73.6	76.3
RI+SSL-Once	57.9	66.7	**69.4**	**71.3**	67.8	78.7	80.5	81.8	65.9	71.6	75.7	78.0
RI+SSL-Twice	60.4	67.6	**69.8**	**71.4**	70.0	79.3	80.7	81.7	66.9	72.2	76.4	78.1
RI+SSL-Twice-SoftLrn	60.6	**68.1**	**69.9**	**71.4**	69.7	79.2	80.5	81.7	66.8	73.0	75.8	78.4
RI+SSL-EM	**62.9**	**68.4**	**69.7**	**71.2**	**70.2**	78.9	80.2	81.3	68.3	71.4	76.4	76.8
CI+No-SSL	44.5	59.2	63.8	69.5	42.1	63.1	71.3	79.6	60.5	68.3	72.8	76.6
CI+SSL-Once	52.5	64.2	67.7	69.6	57.2	75.1	78.3	81.3	60.9	66.7	70.6	74.2
CI+SSL-Twice	54.6	65.2	68.1	69.7	62.4	77.9	80.0	82.2	61.5	66.7	72.1	75.5
CI+SSL-Twice-SoftLrn	52.8	65.3	67.9	70.0	54.7	76.3	79.7	82.3	62.6	67.9	73.1	77.4
CI+SSL-EM	57.4	65.8	68.0	69.6	64.6	78.2	80.2	82.0	62.8	67.0	75.3	77.2

	HepTH				IMDB				PubMed			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	.25%	.5%	1%	3%
RCI+No-SSL	**40.9**	48.1	50.5	**55.0**	41.9	41.9	43.4	43.8	72.6	**77.1**	**80.3**	81.8
RCI+SSL-Once	38.1	48.1	49.8	53.0	41.9	44.3	44.9	45.9	**74.1**	**76.8**	79.0	81.0
RCI+SSL-Twice	37.0	47.6	50.2	52.3	42.2	44.9	45.2	46.0	72.6	74.9	77.2	79.1
RCI+SSL-Twice-SoftLrn	**39.8**	47.8	50.0	52.4	42.4	44.3	44.8	45.0	**72.6**	74.7	77.4	78.8
RCI+SSL-EM	37.2	46.0	48.2	51.2	42.3	45.4	45.4	45.6	67.3	68.9	71.4	72.9
RI+No-SSL	**40.5**	48.3	**51.7**	**55.3**	**45.6**	**48.2**	**48.9**	**50.0**	72.3	**76.4**	79.4	82.5
RI+SSL-Once	**39.9**	**49.9**	**52.4**	**55.4**	**45.4**	47.3	48.0	48.9	**74.3**	**77.3**	79.6	82.1
RI+SSL-Twice	**39.1**	**49.7**	**52.1**	**55.1**	**45.2**	46.9	47.5	48.3	**73.6**	**76.1**	78.4	80.8
RI+SSL-Twice-SoftLrn	**40.2**	**49.9**	**52.6**	**55.0**	44.6	46.3	46.6	46.8	**73.4**	**75.8**	78.5	80.2
RI+SSL-EM	38.0	47.6	49.5	53.2	43.3	46.3	46.4	47.0	67.4	68.8	70.6	73.8
CI+No-SSL	36.8	46.0	48.4	52.6	37.8	38.4	37.6	38.1	68.7	75.7	**80.7**	**82.9**
CI+SSL-Once	26.4	30.0	35.9	43.0	43.1	44.0	44.4	45.3	**73.8**	**77.5**	**80.4**	**83.6**
CI+SSL-Twice	34.1	40.3	42.1	49.6	42.7	45.2	45.7	46.0	**73.2**	76.1	78.6	81.7
CI+SSL-Twice-SoftLrn	37.4	44.6	46.8	50.0	37.5	38.1	38.1	41.8	**73.8**	**77.0**	79.1	81.8
CI+SSL-EM	32.5	41.0	43.8	50.3	37.4	38.9	38.3	37.5	67.8	69.4	72.0	79.1

2, where the number of SSL iterations is one, two, or ten, respectively. SSL-Twice-SoftLrn is SSL-Twice plus soft learning (see Section 5). Finally, No-SSL uses no predicted labels for learning. All methods (including No-SSL) use label regularization (see Section 6.3).

Result: SSL can significantly increase LBC accuracy. SSL-Once consistently increases accuracy for all cases shown with Citeseer, Cora, and Gene (excluding CI for Gene); for RCI and CI with IMDB; and for very sparse graphs with PubMed. The gains can be substantial, and are largest and most consistent when the graph is more sparse. For instance, SSL-Once increases accuracy on Cora by 10.4-15.1% when $d = 1\%$, and by 2.7-7.0% when $d = 5\%$. IMDB (with RI) and HepTH are the notable exceptions; here using SSL-Once does not consistently increase accuracy and sometimes significantly decreases it. For both datasets, SSL struggles because the overall accuracy is low (and thus predicted labels are very noisy). For HepTH the problem is likely exacerbated by the presence of much more autocorrelation than with IMDB (and thus the learned relational features, which are easily influenced by label errors, have much more impact than they do with IMDB).

Result: An appropriate choice of SSL is especially important for CI**.** While some kind of SSL is typically helpful for RCI, RI, and CI, it usually matters most for CI. As discussed in Sections 3 & 7.1, CI's label-based features can be learned only from links that have known (or predicted) labels on both ends of the link. Thus, when no SSL is used, the differences between RI and CI are often especially large (e.g., with Citeseer, Cora, and IMDB); these differences decrease when appropriate SSL is used. Gene is an exception: SSL is less effective here with CI, so SSL tends to increase the advantages of RCI and RI in this case. CI also tends to be more sensitive to the number of SSL iterations: the choice between SSL-ONCE vs. SSL-TWICE vs. SSL-EM matters for all datasets and models, but is especially significant for CI with PubMed (where RCI and RI are also strongly influenced), HepTH, and IMDB.

Result: The optimal number of SSL iterations depends on the dataset and type of model used, but SSL-TWICE **is a reasonable default.** For Citeseer, Cora, and Gene, using more iterations (with SSL-EM) often increases accuracy, and only decreases accuracy by at least 1% compared to SSL-ONCE or SSL-TWICE in one case (for $d = 10\%$ with RI on Gene). With CI on IMDB and with all models on PubMed, however, using SSL-EM substantially decreases accuracy compared to SSL-ONCE and SSL-TWICE. The problem with SSL-EM is that the repeated SSL iterations can sometimes lead to cascading errors that produce highly skewed (and sometimes oscillating) label distributions. Label regularization usually helps to reduce the impact of this problem, but does not eliminate it. For IMDB, the cascading errors are not surprising given its low overall accuracy. PubMed's problems are more surprising, since it has much higher accuracy (although Table IV shows that it has only 3 labels and a default accuracy of 40%, so higher accuracy is less meaningful than with, e.g., Citeseer and Cora). Notably, Zhu et al. [2013] found, for a combined content/link topic prediction model, that PubMed was much more sensitive than datasets such as Cora and Citeseer to the choice of a parameter that controlled the relative importance of "content" (attributes, such as the words in a webpage or publication) vs. links. Thus, the difficulties of PubMed with SSL may reflect the particular challenges of learning appropriate regularization parameters for PubMed (for which we use cross-validation), using only a single sparsely-labeled graph,

Thus, the optimal number of SSL iterations varies depending on the dataset and (sometimes) the choice of RCI, RI, or CI. Fortunately, SSL-TWICE provides relatively strong accuracy in most cases and thus serves a reasonable default to facilitate further comparisons. In particular, while SSL-TWICE is not always the best, it provides accuracy that is almost always within a few points of the best shown in Table V (for each of RCI, RI, and CI), and avoids the severe problems sometimes encountered with SSL-EM. For this reason, we use SSL-TWICE as the default learning method in the rest of this article; Appendix B provides additional results with other choices.[14]

Result: For most datasets, RCI **or** RI **with** SSL-TWICE **yields accuracy that is better than** CI **with any SSL choice.** For Citeseer, Cora, and Gene, using RCI or RI with SSL-TWICE yields accuracy that is, with one exception, better than the accuracy of CI with *any* SSL choice (including NO-SSL) shown in Table V. Similarly, with HepTH and IMDB the same advantage over CI is true for RI. This further illustrates

[14]SSL-ONCE would also be a reasonable default, as used by prior studies [Bilgic et al. 2010; Shi et al. 2011; McDowell and Aha 2013]. However, choosing SSL-ONCE would make some gains of RCI and RI vs. CI appear much larger in Figures 3-5, because CI sometimes has marked accuracy gains with SSL-TWICE vs. with SSL-ONCE. Thus, using SSL-TWICE is more conservative for presenting our results.

the frequent usefulness of neighbor attributes and reasonableness of our default choice of SSL-TWICE.

Result: Soft learning does not consistently increase accuracy with SSL. In Table V, SSL-TWICE-SOFTLRN does not consistently increase accuracy. It increases accuracy vs. SSL-TWICE by at least 1% only for some cases with HepTH and for CI with Gene (and in none of those cases resulting in the overall best accuracy). Notably, with HepTH, many of the "gains" from using soft learning with CI actually only result in increasing the accuracy back to around the same level as NO-SSL, due to HepTH's aforementioned problems with SSL.

In theory, using soft probability estimates (instead of only the most likely labels) should improve learning. In practice, however, if the estimated probabilities are not well calibrated, using them may actually harm learning, as found with CI for IMDB and (sometimes) Cora. The lack of consistent improvement implies that the added complexity of soft learning is not worthwhile for LBC with these datasets. Table VIII in Appendix B shows that these findings generally also hold if soft learning is added to SSL-ONCE or SSL-EM.

8. UNDERSTANDING RELATIONAL AND COLLECTIVE CLASSIFICATION

The previous section found that, for sparsely-labeled networks, relational classification (with RI) often yielded higher accuracy than collective classification (with CI), and that combining these approaches (with RCI) often yielded the best overall accuracy. This section seeks to better understand these differences by answering the following questions:

(1) Why does CI yield lower accuracy than RI only when labels are sparse, and to what extent do CI's struggles here arise from issues with learning vs. with inference?
(2) Compared to CI, RI benefits from being able to use more certain attribute values, instead of predicted labels. How might this change if the attributes were less reliable?
(3) RI with MNAC can often increase accuracy vs. CI. However, would simpler approaches to RI, such as averaging neighbors' attributes, perform just as well?

We address each of these questions in turn. Based on the results of Section 7, we use ICA for inference and SSL-TWICE for learning.

8.1. Loss decomposition analysis of CI vs. RI vs. RCI

In one sense, CI has access to the same "neighbor attribute" information as RI, since CI uses self attributes to predict labels for each node, and then uses those labels as "neighbor labels" for both inference and for semi-supervised learning. Why then does RI generally yield better accuracy when the known labels are sparse? Section 3 (and 7.1) argued that these accuracy differences stem from challenges that CI has with learning *and* inference. Below we examine this argument and seek to better understand the relative importance of these two factors.

Decomposing loss into bias and variance has a long history and can yield insight into the properties of a classification algorithm. Jensen et al. [2004] used such a decomposition for their study of LBC with fully-labeled training graphs, and explained the superior accuracy of CI (vs. RI) in terms of CI's low variance. More recently, Neville and Jensen [2008] showed that LBC loss can be decomposed into four primary terms: learning bias, learning variance, inference bias, and inference variance. They used this method to identify limitations in several LBC methods and propose suitable improvements. However, this decomposition has been rarely used (in part because of the com-

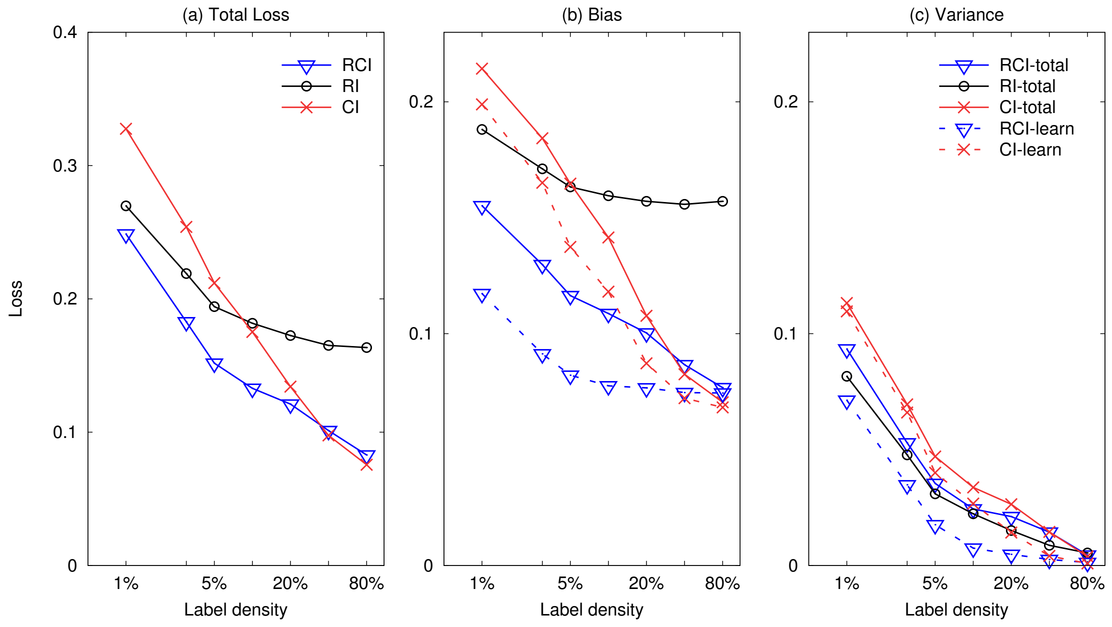

Fig. 7. Bias/variance decomposition on synthetic data with two class labels. Measurements are based on squared loss, so lower values are better. The difference between the “total” and “learn” lines represents bias or variance due to inference.

putational expense of the many trials required) and only with fully-labeled training graphs. We apply it for the first time to sparse, within-network LBC.

In this formulation, *learning* bias and variance are evaluated using a model learned in the standard way, but using “ceiling” inference where the true labels of all neighboring nodes are used for prediction (thus eliminating any need for collective inference). *Inference* bias and variance then measure the difference between those results and the total bias and variance when collective inference is used.

Figure 7 shows the results of this analysis on some synthetic data. We use synthetic data for tractability and because the decomposition only applies to datasets with binary class labels; Appendix A provides details on the data, which was chosen to mimic the characteristics of Cora and Citeseer. The graphs show (a) total loss, (b) learning & total bias, and (c) learning & total variance. For the second two graphs, solid lines indicate the total bias or variance, while dashed lines indicate the amount of bias/variance due to learning. Thus, the gap (if any) between the solid and dashed lines indicates the amount of inference bias or variance; RI has none because it does not use collective inference.

The general shape of Figure 7(a) mimics the behavior shown in Figures 4 & 5 for most datasets: CI outperforms RI (has smaller loss) when the d is high, but RI is better when d is low, and RCI yields accuracy that is comparable or better than both for all values of d. We focus below on the differences between CI and RI.

Result: When the labels are dense, CI **outperforms** RI **because of** CI**’s much lower learning bias.** When d is very high, Figure 7 shows that CI has very little inference bias or variance, and that it outperforms RI (lower total loss) because CI has much lower learning bias. This lower bias is due to a combination of (1) substantial autocorrelation, which makes label-based features highly predictive and (2) an almost fully-labeled network, which facilitates learning based on such features. RI can also learn from all of these labels, but because in this data “self attributes” are only moderately predictive of a node’s label (as with most of the real datasets), then a neighbor’s attributes are helpful for prediction but much less helpful than a neighbor’s label.

Result: When the labels are sparse ($d < 10\%$), RI outperforms CI because of CI’s inference bias and greater learning variance. First, CI’s inference bias results from its use of collective inference. In particular, CI’s inference bias reflects some “flooding” in the network, where sub-regions all get assigned the same label, because the model has learned the presence of autocorrelation and then during inference “same-labeled” regions expand and become self-reinforcing [Sen et al. 2008; Bilgic and Getoor 2008; Xiang and Neville 2011]. Notably, the resultant inference bias is largest when d is between 3% and 10%. With smaller d the model does not learn such strong relational dependencies (because errors in the predicted labels mask the true strength of the autocorrelation), while with larger d there are enough known labels during inference to help reduce flooding (cf., Neville and Jensen 2008). Observe that when $d = 3\%$ or 5%, CI has lower learning bias than RI (see previous paragraph), but higher total bias because of the additional inference bias.[15]

Second, CI has consistently higher learning variance than RI. This variance results from having to learn the parameters for CI’s label-based features using labels that are estimated (and thus contain errors). RI also uses estimated labels during learning, but it has lower learning variance because for each link it needs only one such label for learning (plus attributes), rather than the two labels used by CI. Thus, with RI more of the information used for learning is certain to be correct. Interestingly, Bilgic et al. [2010], Xiang and Neville [2008], and Pfeiffer III et al. [2014b] adopt strategies with CI that choose to learn based *only* on links that involve at least one known label (see Footnote 7), though they did not measure the impact of this choice. That strategy reduces this kind of learning variance for CI, but we previously showed [McDowell and Aha 2012] that learning instead from all available links yields higher accuracy, provided that a hybrid classifier with label regularization is used to improve SSL.

Thus, RI outperforms CI when the labels are sparse due to a combination of reducing learning variance and eliminating all inference bias. Neither of these effects have been previously measured for LBC with sparsely-labeled networks. For instance, Neville and Jensen [2008] showed that *Gibbs* led to high inference variance while belief propagation led to flooding (with high inference bias), but only considered fully-labeled networks. In preliminary work with sparse LBC [McDowell and Aha 2013], we argued that CI lagged RI especially because of difficulties with learning from sparsely-labeled networks, but did not decompose the loss into separate learning and inference components.

Naturally, the magnitude of these trends depends upon the specific data characteristics and especially on the number and kind of labels that are available for learning. Figure 18 in Appendix B considers additional synthetic datasets (with varying amounts of link density and homophily) and finds trends consistent with those reported above.

8.2. The Impact of Noisy or Weakly Predictive Attributes

Compared to CI, RI benefits from being able to use the *attribute* values of neighbors, which are more certain than their predicted *labels*. This eliminates inference bias and reduces learning variance (see Section 8.1), leading to significant performance gains. How might these gains change if the attributes were less reliable?

[15]RCI has even higher inference bias than CI, at low to moderate values of d; this results from (accurately) learning strong relational dependencies, which sometimes magnify errors during collective inference. Despite this inference bias, RCI’s use of neighbor attributes reduces learning bias and helps lead to lower overall loss than with CI or RI.

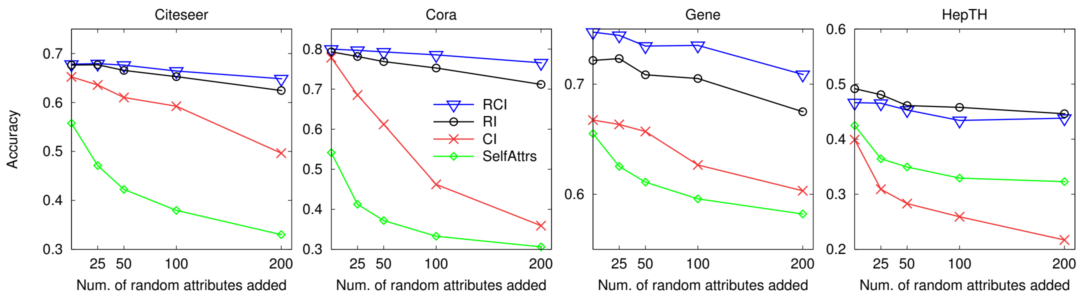

Fig. 8. Accuracy as a number of random attributes are added to the graph and used for prediction, when $d = 3\%$. Accuracy decreases since these attributes are non-informative.

To answer such questions, we examined the real datasets, focusing on the sparse case (i.e., $d = 3\%$) and on the four datasets where LBC had the most utility (Citeseer, Cora, Gene, and HepTH). Figure 8 shows the impact of adding a number of random-valued attributes for each node.

Result: With sparse labeling, RCI **and** RI **better tolerate attribute noise than** CI**, leading to increasing relative accuracy gains for** RCI **and** RI **as noise increases.** Initially, we had expected CI's relative performance to improve as the random attributes were added, based on the results and argument of Jensen et al. [2004]: CI has fewer parameters than RI (see Section 6.2), and thus should suffer less from high variance due to the random attributes. Instead, RI's (and RCI's) gains over CI only increase, for two reasons. First, Jensen et al. had fully-labeled training data, which enabled CI to learn using only true labels. In contrast, for our sparse, semi-supervised setting, CI has greater learning variance than RI because learning the relational parameters typically uses two estimated labels rather than just one (see Section 8.1). Thus, while the addition of noisy attributes adds variance to RI, it adds even more variance to CI, since the estimated labels it uses also depend upon these attributes. Second, Jensen et al. used, for all experiments, a simple Laplace correction for the relational features, whereas we use cross-validation to select regularization parameters based on the data characteristics (see Section 6.2). The regularization reduces variance for RI and CI, and is especially helpful for RI as random attributes are added. If we remove regularization and increase label density, the differences between CI and RI decrease markedly.

For comparison, Figure 8 also shows results for SELFATTRS; accuracy decreases substantially as random attributes are added. With HepTH, the very low accuracy of SELFATTRS leads to cascading errors and flooding with CI, so that CI's accuracy lags that of SELFATTRS's, and this effect gets worse for CI as attribute noise increases. With Citeseer, Cora, and Gene, CI is able to continue to provide some gain over SELFATTRS as the random attributes are added, but in general CI's accuracy decreases markedly as SELFATTRS's accuracy decreases (and sometimes much more sharply, with Cora). In contrast, RCI's and CI's accuracies decrease by substantially less than that of SELFATTRS as attributes are added, showing that RCI and RI are able to effectively leverage the link-based information to compensate for the increased attribute noise.

Result: The advantages of RCI **and** RI **with noisy attributes persist even when alternative sources of attribute noise are considered.** The results described above introduced attribute noise by adding a number of purely random attributes to the model, as done by Jensen et al.'s original study. To ensure that our results were not somehow an artifact of the separation between the two groups of attributes (informative vs. random), we conduct two additional studies. First, Figure 19 (in Appendix

B) displays the results of adding random noise to the *existing* attributes. The results continue to show that RCI and RI tolerate attribute noise better than CI, though (as expected) even their accuracy declines substantially if enough noise is added. Second, recall that most of our datasets use PCA-derived attributes (see Section 6); this simplifies learning and may decrease attribute noise. Thus, Figure 20 (in Appendix B) provides results where we repeat earlier experiments that varied the density d, but now using the unmodified (noisier) attributes. The results show that our use of PCA attributes did in some cases substantially boost accuracy, but using the raw attributes does not change the overall trends previously observed, where RCI and RI are best when d is small, but CI performs at least as well as the others when d is large.

Thus, the diversity of our datasets already showed that using neighbor attributes is often helpful for LBC in a variety of contexts (Figures 4 & 5), and Figures 8 & 19 suggest, surprisingly, that this strategy may be especially important (for sparsely-labeled graphs) when those attributes are weakly predictive or noisy.

8.3. Comparing MNAC vs. Alternative Uses of Neighbor Attributes

MNAC enables the use of neighbor attributes by learning a classifier that predicts a node's label based on the attributes of *one* of its neighbors, then applying this classifier to all neighbors and combining the predictions. This section examines whether MNAC performs better than a natural alternative, and explores a variant of MNAC learning.

The key challenge MNAC addresses is that most classifiers expect each instance to be represented by a fixed-length feature vector, while for LBC each node has a varying number of neighbors. Thus, prior work was either limited to using a classifier that did not require fixed-length feature vectors, or resorted to aggregating (e.g., by averaging or summing) the attribute values for all of a node's neighbors. For instance, if there were 100 attributes indicating the presence of certain words in a webpage, then to handle neighbor attributes we would construct 100 additional features that each represent the average frequency of one of those words amongst a node's neighbors. This approach was used in the early work of Chakrabarti et al. [1998]. Their negative results were highly influential (and further explained by Jensen et al. 2004) but used only a few fully-labeled datasets for training; perhaps in our sparse context this approach might fare better?

Table VI compares MNAC against several variants. Within each section (for RCI or RI), the first row is MNAC as used elsewhere in this article. The second row (MNAC+NoWts) also uses MNAC, but a variant that does *not* weight nodes based on the reciprocal of their degree during learning (see Section 4.3). As an alternative to using MNAC, the third and fourth rows show results that simply use the average of a node's neighbors' attributes as additional features. To be fair to this approach, we considered two variants. NeighAvg-unified uses a single logistic regression model that contains a node's self attributes *and* averaged neighbor attributes; this is analogous to the approach of Chakrabarti et al. [1998]. NeighAvg-hybrid instead uses two logistic regression models, one for self attributes and another for averaged neighbor attributes, using the hybrid model method (see Section 2.1). This allows cross-validation to learn separate regularization constants for self vs. neighbor attributes.

Result: With MNAC**, appropriate node weighting usually improves learning.** In Table VI, using node weighting (with MNAC) instead of using MNAC+NoWts increases accuracy (on average) for all datasets except for Gene (where it has a small negative effect). In particular, the weighting strategy increases accuracy by a little for Citeseer and PubMed, and by substantial amounts for Cora, HepTH, and IMDB. Note that this weighting effectively divides the influence of any single node in the network amongst all of its links, so that the total weight of its links sums to one; this pre-

Table VI. **Alternative uses of neighbor attributes:** Accuracy for different methods of including neighbor attributes as model features for RCI and RI. CI is shown for comparison.

	Citeseer				Cora				Gene			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	1%	3%	5%	10%
RCI+MNAC	61.2	67.8	69.8	71.3	73.3	80.0	81.6	82.9	68.7	74.7	78.1	79.4
RCI+MNAC+NoWts	60.5	68.2	69.4	70.7	68.5	78.2	79.5	81.6	69.0	75.0	77.9	79.0
RCI+NeighAvg-hybrid	59.2	67.4	69.5	71.0	68.9	80.4	81.3	83.2	65.0	71.1	76.5	78.5
RCI+NeighAvg-unified	55.0	65.6	67.9	69.7	62.8	77.9	80.0	82.2	61.4	66.0	70.8	75.4
RI+MNAC	60.4	67.6	69.8	71.4	70.0	79.3	80.7	81.7	66.9	72.2	76.4	78.1
RI+MNAC+NoWts	60.0	67.8	69.2	70.8	65.7	77.7	79.2	81.0	66.9	73.1	76.8	78.5
RI+NeighAvg-hybrid	57.8	67.1	69.5	71.0	62.8	78.0	79.5	81.3	63.1	67.8	72.2	75.1
RI+NeighAvg-unified	48.0	61.2	65.3	68.0	45.3	61.1	65.0	69.9	60.2	64.2	67.6	70.7
CI	54.6	65.2	68.1	69.7	62.4	77.9	80.0	82.2	61.5	66.7	72.1	75.5

	HepTH				IMDB				PubMed			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	.25%	.5%	1%	3%
RCI+MNAC	37.0	47.6	50.2	52.3	42.2	44.9	45.2	46.0	72.6	74.9	77.2	79.1
RCI+MNAC+NoWts	37.5	45.7	47.3	49.6	35.4	37.6	38.7	42.5	70.8	74.8	76.0	78.3
RCI+NeighAvg-hybrid	32.8	42.9	47.6	50.4	42.1	45.0	46.9	46.9	72.5	74.6	77.4	80.1
RCI+NeighAvg-unified	33.5	41.0	42.0	49.2	41.6	45.8	46.9	48.5	73.9	75.7	78.8	81.4
RI+MNAC	39.1	49.7	52.1	55.1	45.2	46.9	47.5	48.3	73.6	76.1	78.4	80.8
RI+MNAC+NoWts	37.5	45.2	46.9	49.2	40.9	39.8	39.3	38.2	71.5	75.9	77.5	80.2
RI+NeighAvg-hybrid	37.0	48.1	50.9	55.1	44.9	48.5	48.9	49.8	72.5	75.4	78.7	81.5
RI+NeighAvg-unified	32.2	42.1	44.6	48.1	43.6	47.0	47.9	49.2	70.6	75.0	77.8	81.4
CI	34.1	40.3	42.1	49.6	42.7	45.2	45.7	46.0	73.2	76.1	78.6	81.7

vents a single label error from having widespread impact during learning. However, high degree nodes continue to exert larger influence during inference, where no such weighting is performed. Note that Rattigan et al. [2007] and Bilgic and Getoor [2008] found that high degree nodes were not especially helpful to label for "active inference"; this may suggest that in our context extending the "inverse degree" weighting idea to inference could be useful future work.

Result: For sparse networks, MNAC yields higher accuracy than NeighAvg for most datasets. Table VI shows that MNAC generally yields higher accuracy than either variant of NeighAvg when d is small. In particular, with RI, MNAC performs better than NeighAvg for Citeseer, Cora, Gene, and HepTH for all $d \leq 5\%$ and almost always when $d = 10\%$. These gains are substantial in some cases, e.g. by more than 7% for Cora when d=1% and by 3%-4.4% for Gene when $d \leq 10\%$. With RCI, the same gains are generally present but somewhat smaller due to the helpful inclusion of neighbor labels. For IMDB and PubMed, the choice of MNAC vs. NeighAvg matters much less, due to the relatively small impact of LBC in general on these datasets. For these datasets NeighAvg sometimes leads to higher accuracy, though the differences are generally small.

Result: Using a hybrid classifier with NeighAvg substantially increases accuracy. Table VI shows that NeighAvg-hybrid consistently outperforms NeighAvg-unified, sometimes by more than 5%, for all cases except for RCI with IMDB and PubMed (and once with HepTH). Such hybrid classifiers have been rarely used in prior LBC studies, but these gains are consistent with the few prior studies where such comparisons were reported [McDowell and Aha 2012] or alluded to [Lu and Getoor 2003b]. Note that our inclusion of NeighAvg-hybrid makes our results more conservative; if

MNAC were compared only against NeighAvg-unified then MNAC's gains would look substantially larger.

9. RELATED WORK

This section describes additional related work, focusing especially on alternate methods for performing LBC in sparsely-labeled networks.

Latent and walk-based methods for LBC: Section 3 described how label sparsity presents a number of challenges that affect LBC learning and inference with CI. Sections 4-5 considered how appropriate use of neighbor attributes and semi-supervised learning could possibly help address these challenges (and Appendix B considers improved inference). In each case, however, we kept the general paradigm of traditional LBC methods: a classifier is learned from the provided attributes and labels of the network, which is then applied with collective or relational inference to generate final label predictions.

A number of recent studies have also recognized the challenges inherent in sparsely-labeled LBC, but instead proposed more fundamental changes to learning and/or inference. In particular, some researchers have argued that label sparsity makes learning from the original attributes and links of the network ineffective, and thus advocate constructing a new set of "latent" features or links that simplify learning and/or inference.

We first consider latent methods that use only the network link structure as inputs, ignoring all attributes and labels during latent feature/link construction. One early approach to this idea was the "ghost edges" method of Gallagher et al. [2008]. They use repeated random walks to quantify, for each unlabeled node, its proximity to every labeled node. Each such pair then gets a new link in the graph, with weight based on the measured proximity. They then use the new links to classify each node using either (a) wvRN (see Section 2) or (b) a supervised classifier in conjunction with ICA. They report higher accuracy compared to several competitors, but only consider tasks where no attributes are available. For our webpage example, this would be akin to classifying pages based only on their links and a set of known labels.

Other methods use the graph link structure not to create more links but to create latent features for each node. For instance, Tang and Liu [2011] perform spectral clustering on the links to extract latent features, while Tang et al. [2011] use a scalable k-means clustering of the links to produce latent features that are certain to be sparse (e.g., having few non-zero values). In each case, the new features are provided to a link-unaware supervised classifier, where learning uses only the labeled nodes and their new latent features.

The above methods all generate latent features or links in an unsupervised manner, e.g., ignoring the provided labels until learning or inference time. In contrast, Menon and Elkan [2010] use an approach similar to that of Tang and Liu [2009], but where the known labels influence the latent feature construction. They report mixed results for the impact on accuracy. Shi et al. [2011] use the provided labels, and also the node attributes. Their goal is to construct new latent links so that the modified network has high homophily (i.e., nodes are very likely to link to other nodes with the same label), so that a simple inference procedure can then be used. In particular, they generate five fully-connected "latent graphs" where link weights in each graph are based on a quantity such as attribute similarity or proximity in the original graph. They then use quadratic programming to identify an optimal set of weights for combining these five graphs into a single final "output" graph so that homophily among the known labels is maximized. Finally, they use a simple form of label propagation, similar to wvRN, on the output graph to predict labels using the new links.

Other work uses matrix factorization in some form to produce new latent features [Zhu et al. 2007; Hoff 2009; Miller et al. 2009] using a supervised or unsupervised method. They typically have used smaller datasets; scalability is a challenge for some of them [Menon and Elkan 2011].

Finally, some researchers have argued that LBC learning is fundamentally difficult when only a few known labels are available, and thus advocated the use of a non-learning method. Specifically, Lin and Cohen [2010] propose "MultiRankWalk," a "neighbor labels only" method that uses multiple random walks, each starting from a known label, to predict labels for the unknown nodes. They find that this method yields higher accuracy than WVRN, another non-learning, "neighbor labels only" method, when the labels are very sparse. These variants can increase accuracy, but only for graphs that match their assumptions (the presence of significant homophily, rather than more complex forms of autocorrelation) and have enough known labels. Recently, Neumann et al. [2013] proposed a new method based on coinciding walk kernels. They showed that these kernels could increase accuracy even for datasets where simple homophily was not present. This method, like MultiRankWalk and WVRN, ignores attribute values. Finally, Wang and Sukthankar [2013] consider how to extend WVRN to the case where nodes may be assigned to multiple labels.

In recent work [Fleming et al. 2014; McDowell et al. 2015], we evaluate many of the methods described above, including those of Tang et al., Shi et al., and Lin and Cohen. Compared to RCI, RI, and CI, we found that some of the the latent methods can be competitive when the network is densely-labeled or when the attributes are not very informative, but that when the network is sparsely-labeled, RCI and/or RI generally provide the best accuracy, with some significant gains vs. the latent methods.

Additional methods for LBC: In this article we used the pseudolikelihood for learning (see Equation 3 in Section 2.1), as did (explicitly or implicitly) all of the prior studies shown in Table III. Learning could also be accomplished by maximizing the joint likelihood (Equation 2), for instance using some form of Markov network, such as Relational Markov Networks (RMNs) [Taskar et al. 2002], Markov Logic Networks (MLNs) [Richardson and Domingos 2006], or Hinge-loss Markov random fields (HL-MRFs) [Bach et al. 2013].[16] In this domain, inference methods include loopy belief propagation [Taskar et al. 2002], Markov chain Monte Carlo (MCMC) [Richardson and Domingos 2006], and MaxWalkSAT [Domingos et al. 2008], amongst others. Markov networks can elegantly represent rich dependencies, and could be modified to handle more complex tasks (such as joint link and label prediction). Some work, however, has found it difficult to use them to achieve high LBC accuracy compared to approaches based on CI [Sen et al. 2008; Neville and Jensen 2008; Crane and McDowell 2012]. On the other hand, there are multiple possible ways of improving their performance for LBC. Domingos and Lowd [2009] report competitive accuracies for MLNs executing LBC for the "WebKB" dataset, obtained by more careful construction of the logical rules that specify the autocorrelation patterns to learn. Moreover, many recent papers have studied improving learning and/or inference for these Markov networks. One common theme has been improving inference by exploiting symmetries in the network (e.g., Ahmadi et al. 2013; Singla et al. 2014) and/or by applying some restriction on the network structure or potential functions that can be used [Torkamani and Lowd 2013; Bach et al. 2015]. See Kimmig et al. [2015] for a useful survey that also discusses learning. Future work should evaluate the impact of using these methods on sparse LBC, with a variety of datasets.

[16]While these formalisms support learning based on the joint likelihood, pseudolikelihood is often a practical alternative, for instance as used by some work with RMNs [Bilgic et al. 2010; Namata et al. 2011].

For semi-supervised learning (SSL), we focus on the "self learning" paradigm, where model predictions on unlabeled data are used for subsequent model learning steps [Chapelle et al. 2006]. This is a common choice (e.g., used by all the work shown in Table III), but other approaches are possible. For instance, some SSL methods transform non-relational data to a graph or network (e.g., by linking "similar" instances), then use manifold regularization [Belkin et al. 2006; Geng et al. 2012], low-density separation assumptions (e.g., via a transductive support vector machine) [Vapnik 1998], harmonic minimization [Zhu et al. 2003], local global consistency [Zhou et al. 2004], or label propagation [Zhu and Ghahramani 2002; Wang and Zhang 2008]. Such methods could also be applied to our datasets, where the data is already represented as a network, and future work should compare against some of these approaches. Notably, Macskassy and Provost [2007] demonstrated that their WVRN method was closely related to the harmonic minimization approach of Zhu et al. (and produced identical accuracy); Appendix B presents results with WVRN.

Most of the SSL methods discussed in the preceding paragraph link "similar" nodes together because of an implicit smoothness assumption: similar instances are likely to have the same label. Thus, unlike the models for RCI, CI, and RI that we studied in this article, they do not naturally extend well to datasets that have useful relational correlations but do *not* have high homophily (see discussion above, in context of "MultiRankWalk", of homophily vs. alternative types of correlation). In contrast, Dhurandhar and Wang [2013] show how to extend some such SSL methods to support more complex types of autocorrelation. Specifically, they construct edge weights for the graph, where weights may be positive or negative, and are influenced by the known labels as well by the similarity between nodes' attributes (if present). They use these weights to predict labels using a modified form of graph Laplacian regularization (e.g., as used by Zhou et al.). They show that this method can boost accuracy, compared to several existing LBC methods, on several partially-labeled datasets that exhibit more complex linking patterns, but do not consider any LBC methods that exploit neighbor attributes.

Sen et al. [2008] discusses other possible approaches to LBC such as linear programming and graph cut-based methods. Gould and He [2014] provide a useful survey of a form of LBC applied to labeling inter-related superpixels in images. They assume a RMN-like formalism and report impressive accuracy results, though many of the algorithms depend upon a particular graph structure (e.g., a pixel is linked only to its four immediate neighbors in the image), rather than supporting arbitrary links as in the models used in this article.

10. SIGNIFICANCE AND IMPLICATIONS FOR FUTURE WORK

To conclude, we now revisit the significance and novelty of some of our contributions (see also the list of Section 1), and examine their implications for future work.

Discovering and enabling neighbor attributes as features for LBC: We demonstrated, for the first time, how using neighbor attributes could often significantly increase LBC accuracy with sparsely-labeled networks. In hindsight, should we have expected this result? Conceivably, one could argue that our results should not be surprising – after all, if few neighbor *labels* are available for learning and inference, isn't it logical to expect models that include neighbor *attributes* to attain higher accuracy? On the other hand, if the labeled data is scarce, perhaps adding more attributes might actually decrease accuracy, by causing the model to overfit?

Therefore, the usefulness of neighbor attributes for LBC was not a foregone conclusion (and indeed prior work had argued against using them). Moreover, no prior work had (1) demonstrated that neighbor attributes can improve accuracy on any synthetic or real datasets, (2) evaluated the impact of different data characteristics on the relative accuracy of such models, or (3) demonstrated how neighbor attributes may be used with discriminative classifiers. Additionally, almost no recent work has considered neighbor attributes for these problems, even for the most relevant case where labels are sparse (see Table III). We argued in Section 4 that neighbor attributes have been ignored because they have long been thought unhelpful (based on prior work) and because they fit poorly with the classifiers that were otherwise deemed most appropriate (e.g., logistic regression, as in most studies shown in Table III, or support vector machines as used in other recent work). Thus, the primary results of this article—demonstrating the usefulness of neighbor attributes for classification in sparse networks, and how the new MNAC method can exploit such attributes with discriminative classifiers—have not been presented before, are not widely accepted or used, and can be expected to have a significant impact on future research.

In particular, these results have specific implications for the evaluation of past and future research. For instance, Bilgic et al. [2010] studied active learning with sparsely-labeled graphs using only CI. For this problem, the optimal label acquisition strategies could be quite different if RI or RCI were considered, since they could tolerate learning with fewer and more widely-dispersed labels [McDowell 2015]. Similar changes, if neighbor attributes were included, may apply to other recent findings related to active exploration [Pfeiffer III et al. 2014a] and active surveying [Namata et al. 2012] in networks. Likewise, Shi et al. [2011] used ICA with CI as a baseline, and found that CI performed poorly compared to their new "latent network propagation" method, but did not consider the use of neighbor attributes. However, comparing latent network propagation's results from McDowell et al. [2015] (which are directly comparable with those in this article) with Table V reveals that RI outperforms latent network propagation on all six datasets, *even if* RI *uses no SSL*. Thus, using a simple version of RI, with MNAC, as a baseline would have substantially changed their conclusions about the benefits of their proposed method. Overall, our findings should encourage future researchers to consider neighbor attributes in models for LBC, to include RI and RCI as baselines in comparisons, and to re-evaluate some earlier work (see Table III) that did not consider such models.

Examining why CI **and** RI **behave as they do:** We also examined *why* RI often, but not always, yields higher accuracy than CI. For instance, neighbor attributes are not inherently better for prediction; indeed, Figures 3-5 and Table V show that, given appropriate conditions, using neighbor labels instead (with CI) may be best. More specifically, Section 8 discussed how using neighbor attributes specifically improves learning variance and inference bias under certain conditions. We also showed that these advantages may actually grow when the attributes are noisy—the opposite of previous findings with fully-labeled networks.

Understanding and improving SSL for LBC: This article is not the first to apply SSL methods to sparsely-labeled LBC, but it is the first to systematically compare multiple SSL variants and to demonstrate consistent gains for RCI, RI, and CI. Specifically, we explained how almost all prior SSL methods from the studies of Table III can be described as specific instances of one general SSL algorithm. We showed that the number of SSL learning iterations used could have a substantial impact on the results, while the use of soft learning typically did not. Moreover, Section 7.3 showed that a single iteration of SSL, as used by many prior studies [Bilgic et al. 2010; Shi et al. 2011; McDowell and Aha 2013], may not be enough to adequately leverage the link-based correlations, although more SSL iterations is not always better. Thus, some of those earlier studies may need to be re-visited.

Discussion & Limitations: Compared to related work (cf., Table III), we evaluated at least as many real datasets, in addition to considering a range of learning choices, inference algorithms (see Appendix B), and data characteristics. Nonetheless, our results should be confirmed with additional datasets, model types, and inference algorithms, and the best methods should be compared against others discussed in Section 9. In addition, we assumed that all links in the network were known, and that the model structure was the same as the network structure. Future work should consider the impact of including structure learning [Neville et al. 2003b; Kok and Domingos 2005; Huynh and Mooney 2008] and link prediction [Liben-Nowell and Kleinberg 2007; Namata et al. 2011] in conjunction with LBC, as well as the usefulness of other types of links (such as co-citation links) [Macskassy and Provost 2007; McDowell et al. 2009] for some datasets.

Future work should also consider if RI's and RCI's learning variance could be further reduced, perhaps by learning multiple models via some form of relational resampling [Kuwadekar and Neville 2010]. In addition, creating novel feature selection methods to automatically choose the appropriate mix of features based on neighbor labels and/or attributes would be useful, possibly leveraging recent work for doing so when few if any labels are known (cf., Tang and Liu 2012). Finally, there are other types of networks beyond the homogeneous (single node-type) networks that we have considered here, and in some contexts (such as social networks), some attribute values may be missing due to incomplete profiles or privacy restrictions. Thus, we intend to explore LBC with neighbor attributes in social networks and other more heterogeneous networks that include nodes or links of different types (cf., Leskovec et al. 2010; Kong et al. 2012), some missing attribute values, and/or where nodes may have multiple true labels.

APPENDIX

A. DETAILS ON THE SYNTHETIC DATA

We used a synthetic data generator with two components: a Graph Generator and an Attribute Generator. The Graph Generator has four inputs: N_N (the number of nodes), N_C (the number of class labels), ld (the link density), and dh (the degree of homophily). For each link, dh controls the probability that the linked nodes have the same class label. Higher values of dh yield higher autocorrelation, and the value of dh is roughly equivalent to the *label consistency* metric shown in Table IV. Link generation depends on the idea of *preferential attachment*, where a new link "prefers" to attach to existing nodes that have higher degree. The final number of links is approximately $N_N/(1-ld)$, and the final link degrees follow a power law distribution, which is common in real networks [Bollobás et al. 2003]. The Graph Generator is identical to that used by Sen et al. [2008]; see that article for more detail. We use $N_C = 2$ (as required by the bias/variance decomposition, Neville and Jensen 2008), and $N_N = 1000$. By default, we selected $ld = 0.4$ and $dh = 0.7$. The Attribute Generator is identical to that used by McDowell et al. [2009]. It creates ten binary attributes with a default *attribute predictiveness* (ap) of 0.3. Together, these choices result in graphs with characteristics similar to Cora and Citeseer; Figure 18 (in Appendix B) examines results with other choices.

All results are averaged over 25 trials. As with the real data, we used cross-validation to select appropriate regularization parameters.

B. ADDITIONAL DESCRIPTIONS AND RESULTS (ONLINE APPENDIX B)

To facilitate comparison, we used default choices for the performance metric (accuracy), underlying classifier (logistic regression), inference method (ICA), and learning algorithm (SSL-TWICE). An online "Appendix B" presents results with additional choices and additional synthetic data, and is posted along with the online version of this article.

As mentioned in Section 5, Appendix B also includes a fuller description of ICA, descriptions and experimental comparisons for alternative inference methods, and a discussion of computational complexity.

REFERENCES

B. Ahmadi, K. Kersting, M. Mladenov, and S. Natarajan. 2013. Exploiting symmetries for scaling loopy belief propagation and relational training. *Machine Learning* 92, 1 (2013), 91–132.

S. H. Bach, B. Huang, and L. Getoor. 2015. Unifying Local Consistency and MAX SAT Relaxations for Scalable Inference with Rounding Guarantees. In *Proc. of AISTATS*. 46–55.

S. H. Bach, B. Huang, B. London, and L. Getoor. 2013. Hinge-loss Markov Random Fields: Convex Inference for Structured Prediction. In *Proc. of UAI*.

M. Belkin, P. Niyogi, and V. Sindhwani. 2006. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *The Journal of Machine Learning Research* 7 (2006), 2399–2434.

J. Besag. 1975. Statistical analysis of non-lattice data. *The statistician* (1975), 179–195.

J. Besag. 1986. On the statistical analysis of dirty pictures. *Journal of the Royal Statistical Society* 48, 3 (1986), 259–302.

M. Bilgic and L. Getoor. 2008. Effective Label Acquisition for Collective Classification. In *Proc. of KDD*. 43–51.

M. Bilgic, L. Mihalkova, and L. Getoor. 2010. Active Learning for Networked Data. In *Proc. of ICML*. 79–86.

B. Bollobás, C. Borgs, J. Chayes, and O. Riordan. 2003. Directed scale-free graphs. In *Proc. of SODA*. 132–139.

S. Chakrabarti, B. Dom, and P. Indyk. 1998. Enhanced hypertext categorization using hyperlinks. In *Proc. of SIGMOD*. 307–318.

O. Chapelle, B. Schölkopf, and A. Zien. 2006. *Semi-supervised learning*. MIT Press Cambridge.

R. Crane and L. McDowell. 2012. Investigating Markov Logic Networks for Collective Classification.. In *Proc. of ICAART*. 5–15.

A. P. Dempster, N. M. Laird, and D. B. Rubin. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B* (1977), 1–38.

A. Dhurandhar and J. Wang. 2013. Single Network Relational Transductive Learning. *Journal of Artificial Intelligence Research* 48 (2013), 813–839.

P. Domingos, S. Kok, D. Lowd, H. Poon, M. Richardson, and P. Singla. 2008. *Probabilistic Inductive Logic Programming: Theory and Applications*. Springer Berlin Heidelberg, Chapter Markov Logic, 92–117.

P. Domingos and D. Lowd. 2009. Markov logic: An interface layer for artificial intelligence. *Synthesis Lectures on Artificial Intelligence and Machine Learning* 3, 1 (2009), 1–155.

A. Fleming, L. K. McDowell, and Z. Markel. 2014. A Hidden Treasure? Evaluating and Extending Latent Methods for Link-based Classification. In *Proc. of IRI*. 669–676.

B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos. 2008. Using ghost edges for classification in sparsely labeled networks. In *Proc. of KDD*. 256–264.

B. Geng, D. Tao, C. Xu, L. Yang, and X.-S. Hua. 2012. Ensemble manifold regularization. *Pattern Analysis and Machine Intelligence, IEEE Transactions on* 34, 6 (2012), 1227–1233.

S. Gould and X. He. 2014. Scene understanding by labeling pixels. *Commun. ACM* 57, 11 (2014), 68–77.

D. J. Hand and R. J. Till. 2001. A Simple Generalisation of the Area Under the ROC Curve for Multiple Class Classification Problems. *Machine Learning* 45, 2 (2001), 171–186.

P. Hoff. 2009. Multiplicative latent factor models for description and prediction of social networks. *Computational & Mathematical Organization Theory* 15, 4 (2009), 261–272.

T. N. Huynh and R. J. Mooney. 2008. Discriminative structure and parameter learning for Markov logic networks. In *Proc. of ICML*. ACM, 416–423.

Y. Jacob, L. Denoyer, and P. Gallinari. 2014. Learning Latent Representations of Nodes for Classifying in Heterogeneous Social Networks. In *Proc. of WSDM*. 373–382.

D. Jensen and J. Neville. 2002. Linkage and Autocorrelation Cause Feature Selection Bias in Relational Learning. In *Proc. of ICML*. 259–266.

D. Jensen, J. Neville, and B. Gallagher. 2004. Why Collective Inference Improves Relational Classification. In *Proc. of KDD*. 593–598.

A. Kimmig, L. Mihalkova, and L. Getoor. 2015. Lifted graphical models: A survey. *Machine Learning* 99, 1 (2015), 1–45.

S. Kok and P. Domingos. 2005. Learning the structure of Markov logic networks. In *Proc. of ICML*. ACM, 441–448.

X. Kong, P. S. Yu, Y. Ding, and D. J. Wild. 2012. Meta path-based collective classification in heterogeneous information networks. In *Proc. of CIKM*. 1567–1571.

A. Kuwadekar and J. Neville. 2010. Combining Semi-supervised Learning and Relational Resampling for Active Learning in Network Domains. In *Proc. of the Budgeted Learning Workshop at ICML-2010*.

A. Kuwadekar and J. Neville. 2011. Relational active learning for joint collective classification models. In *Proc. of ICML*. 385–392.

J. Leskovec, D. Huttenlocher, and J. Kleinberg. 2010. Predicting Positive and Negative Links in Online Social Networks. In *Proc. of WWW*. 641–650.

D. Liben-Nowell and J. Kleinberg. 2007. The link-prediction problem for social networks. *Journal of the American Society for Information Science and Technology* 58, 7 (2007), 1019–1031.

F. Lin and W. W. Cohen. 2010. Semi-supervised classification of network data using very few labels. In *Proc. of ASONAM*. 192–199.

Q. Lu and L. Getoor. 2003a. Link-based Classification. In *Proc. of ICML*. 496–503.

Q. Lu and L. Getoor. 2003b. Link-based Classification using Labeled and Unlabeled Data. In *ICML Workshop on the Continuum from Labeled to Unlabeled data*.

S. Macskassy and F. Provost. 2007. Classification in Networked Data: A toolkit and a univariate case study. *Journal of Machine Learning Research* 8 (2007), 935–983.

L. K. McDowell. 2015. Relational Active Learning for Link-Based Classification. In *Proc. of DSAA*.

L. K. McDowell and D. W. Aha. 2012. Semi-Supervised Collective Classification via Hybrid Label Regularization. In *Proc. of ICML*. 975–982.

L. K. McDowell and D. W. Aha. 2013. Labels or attributes? Rethinking the neighbors for collective classification in sparsely-labeled networks. In *Proc. of CIKM*. 847–852.

L. K. McDowell, A. Fleming, and Z. Markel. 2015. Evaluating and Extending Latent Methods for Link-based Classification. In *Advances in Intelligent Systems and Computing*, Vol. 346. Springer International Publishing, 227–256.

L. K. McDowell, K. M. Gupta, and D. W. Aha. 2009. Cautious collective classification. *Journal of Machine Learning Research* 10 (2009), 2777–2836.

A. Menon and C. Elkan. 2010. Predicting labels for dyadic data. *Data Mining and Knowledge Discovery* 21, 2 (2010), 327–343.

A. Menon and C. Elkan. 2011. Link prediction via matrix factorization. *Machine Learning and Knowledge Discovery in Databases* (2011), 437–452.

K. Miller, T. Griffiths, and M. Jordan. 2009. Nonparametric latent feature models for link prediction. *Advances in Neural Information Processing Systems (NIPS)* (2009), 1276–1284.

G. Namata, S. Kok, and L. Getoor. 2011. Collective graph identification. In *Proc. of KDD*. 87–95.

G. M. Namata, B. London, L. Getoor, and B. Huang. 2012. Query-driven Active Surveying for Collective Classification. In *Proc. of the 10th International Workshop on Mining and Learning with Graphs (MLG-2010)*.

M. Neumann, R. Garnett, and K. Kersting. 2013. Coinciding Walk Kernels: Parallel Absorbing Random Walks for Learning with Graphs and Few Labels. In *Proc. of ACML*. 357–372.

J. Neville and D. Jensen. 2000. Iterative Classification in Relational Data. In *Proc. of the Workshop on Learning Statistical Models from Relational Data at AAAI-2000*. 13–20.

J. Neville and D. Jensen. 2005. Leveraging Relational Autocorrelation with Latent Group Models. In *Proc. of ICDM*. 170–177.

J. Neville and D. Jensen. 2007. Relational Dependency Networks. *Journal of Machine Learning Research* 8 (2007), 653–692.

J. Neville and D. Jensen. 2008. A Bias/Variance Decomposition for Models Using Collective Inference. *Machine Learning Journal* 73, 1 (2008), 87–106.

J. Neville, D. Jensen, L. Friedland, and M. Hay. 2003b. Learning Relational Probability Trees. In *Proc. of KDD*. 625–630.

J. Neville, D. Jensen, and B. Gallagher. 2003a. Simple Estimators for Relational Bayesian Classifiers. In *Proc. of ICDM*. 609–612.

J. J. Pfeiffer III, J. Neville, and P. N. Bennett. 2014a. Active Exploration in Networks: Using Probabilistic Relationships for Learning and Inference. In *Proc. of CIKM*. 639–648.

J. J. Pfeiffer III, J. Neville, and P. N. Bennett. 2014b. Composite Likelihood Data Augmentation for Within-Network Statistical Relational Learning. In *Proc. of ICDM*. 490–499.

J. J. Pfeiffer III, J. Neville, and P. N. Bennett. 2015. Overcoming Relational Learning Biases to Accurately Predict Preferences in Large Scale Networks. In *Proc. of WWW*. 853–863.

M. J. Rattigan, M. Maier, D. Jensen, B. Wu, X. Pei, J. Tan, and Y. Wang:. 2007. Exploiting Network Structure for Active Inference in Collective Classification. In *Proc. of the Workshop on Mining Graphs and Complex Structures at ICDM-2007*. 429–434.

M. Richardson and P. Domingos. 2006. Markov Logic Networks. *Machine Learning* 62, 1-2 (2006), 107–136.

R. A. Rossi, L. K. McDowell, D. W. Aha, and J. Neville. 2012. Transforming Graph Data for Statistical Relational Learning. *Journal of Artificial Intelligence Research* 45 (2012), 363–441.

T. Saha, H. Rangwala, and C. Domeniconi. 2014. FLIP: Active Learning for Relational Network Classification. In *Proc. of ECML PKDD*. 1–18.

P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad. 2008. Collective Classification in Network Data. *AI Magazine* 29, 3 (2008), 93–106.

X. Shi, Y. Li, and P. Yu. 2011. Collective Prediction with Latent Graphs. In *Proc. of CIKM*. 1127–1136.

P. Singla, A. Nath, and P. Domingos. 2014. Approximate lifting techniques for belief propagation. In *Proc. of AAAI*. 2497–2504.

J. Tang and H. Liu. 2012. Unsupervised feature selection for linked social media data. In *Proc. of KDD*. 904–912.

L. Tang and H. Liu. 2009. Relational learning via latent social dimensions. In *Proc. of KDD*. 817–826.

L. Tang and H. Liu. 2011. Leveraging social media networks for classification. *Data Mining and Knowledge Discovery* 23, 3 (2011), 447–478.

L. Tang, X. Wang, and H. Liu. 2011. Scalable Learning of Collective Behavior. *IEEE Transactions on Knowledge and Data Engineering* 24 (2011), 1080–1091. Issue 6.

B. Taskar, P. Abbeel, and D. Koller. 2002. Discriminative Probalistic Models for Relational Data. In *Proc. of UAI*. 485–492.

B. Taskar, E. Segal, and D. Koller. 2001. Probabilistic classification and clustering in relational data. In *Proc. of IJCAI*. 870–878.

M. Torkamani and D. Lowd. 2013. Convex Adversarial Collective Classification. In *Proc. of the 30th International Conference on Machine Learning (ICML)*. 642–650.

V. N. Vapnik. 1998. *Statistical learning theory*. Wiley New York.

F. Wang and C. Zhang. 2008. Label propagation through linear neighborhoods. *Knowledge and Data Engineering, IEEE Transactions on* 20, 1 (2008), 55–67.

T. Wang, J. Neville, B. Gallagher, and T. Eliassi-Rad. 2011. Correcting Bias in Statistical Tests for Network Classifier Evaluation. In *Proc. of ECML*. 506–521.

X. Wang and G. Sukthankar. 2013. Multi-label relational neighbor classification using social context features. In *Proc. of KDD*. 464–472.

R. Xiang and J. Neville. 2008. Pseudolikelihood EM for within-network relational learning. In *Proc. of ICDM*. 1103–1108.

R. Xiang and J. Neville. 2011. Understanding propagation error and its effect on collective classification. In *Proc. of ICDM*. 834–843.

D. Zhou, O. Bousquet, T. N. Lal, J. Weston, and B. Schölkopf. 2004. Learning with local and global consistency. *Advances in Neural Information Processing Systems (NIPS)* 16 (2004), 321–328.

S. Zhu, K. Yu, Y. Chi, and Y. Gong. 2007. Combining content and link for classification using matrix factorization. In *Proc. of SIGIR*. 487–494.

X. Zhu and Z. Ghahramani. 2002. *Learning from labeled and unlabeled data with label propagation*. Technical Report CMU-CALD-02-107. Cargenie Mellon University.

X. Zhu, Z. Ghahramani, and J. D. Lafferty. 2003. Semi-Supervised Learning Using Gaussian Fields and Harmonic Functions. In *Proc. of ICML*. 912–919.

Y. Zhu, X. Yan, L. Getoor, and C. Moore. 2013. Scalable Text and Link Analysis with Mixed-topic Link Models. In *Proc. of KDD*. 473–481.