Correcting Relational Bias to Improve Classification in Sparsely-Labeled Networks

Joshua R. King
Department of Computer Science
U.S. Naval Academy
Annapolis, Maryland 21402
Email: jrkingjr9@gmail.com

Luke K. McDowell
Department of Computer Science
U.S. Naval Academy
Annapolis, Maryland 21402
Email: lmcdowel@usna.edu

***Abstract*—Many classification problems involve nodes that have a natural connection between them, such as links between people, pages, or social network accounts. Recent work has demonstrated how to learn relational dependencies from these links, then leverage them as predictive features. However, while this can often improve accuracy, the use of linked information can also lead to cascading prediction errors, especially in the common-case when a network is only sparsely-labeled. In response, this paper examines several existing and new methods for correcting the "relational bias" that leads to such errors. First, we explain how existing approaches can be divided into "resemblance-based" and "assignment-based" methods, and provide the first experimental comparison between them. We demonstrate that all of these methods can improve accuracy, but that the former type typically leads to better accuracy. Moreover, we show that the more flexible methods typically perform best, motivating a new assignment-based method that often improves accuracy vs. a more rigid method. In addition, we demonstrate for the first time that some of these methods can also improve accuracy when combined with Gibbs sampling for inference. However, we show that, with Gibbs, correcting relational bias also requires improving label initialization, and present two new initialization methods that yield large accuracy gains. Finally, we evaluate the effects of relational bias when "neighbor attributes," recently-proposed additions that can provide more stability during inference, are included as model features. We show that such attributes reduce the negative impact of bias, but that using some form of bias correction remains important for achieving maximal accuracy.**

I. Introduction

Many problems in communications, social networks, biology, business, etc. involve classifying nodes in a network (or *graph*). For instance, consider predicting a class label for each page (node) in a set of linked webpages, where some 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) [1], [2] also uses, for each node, the attributes or labels of *neighboring* pages as model features. For instance, to predict node v's label, a classifier might use v's attributes along with features based on the fraction of v's neighbors that have a positive label. This will require some kind of iterative *collective inference*, since many such labels will initially be unknown [3], [4]. Applications of LBC have included, for instance, document topic prediction [4], biological discovery [3], and movie revenue estimation [5].

Most early work on link-based classification assumed a fully-labeled training graph. However, often (e.g., for social and webpage networks) collecting the node attributes and link structure for this graph may be easy, but acquiring the desired labels can be much more expensive [6], [7]. In response, recent studies have examined LBC methods with partially-labeled training graphs, using some form of semi-supervised learning (SSL) to leverage the unlabeled portion of the graph [7], [8], [9].

While LBC has been shown to be effective in many contexts, the use of *relational* (link-based) information as predictive features is not without its dangers, some of which may be amplified by the use of SSL. First, collective inference sometimes leads to "over-propagation" (or "flooding") [10], [4], [11], where inference ends up assigning the same label to large regions of the network. Over-propagation may occur due to a combination of (a) learned label correlations favoring situations where most links connect two nodes with the same (predicted or known) label and (b) relational features dominating during inference. Second, repeated SSL iterations may also create self-feedback loops that cause some labels to be over represented in the predictions, leading to the learning of less accurate models in the next SSL iteration.

We refer to both of these problems as issues of "relational bias" (cf., [12]), and they can lead to large decreases in accuracy [4], [9], [12]. Prior work has sought to address this bias in several ways. First, some methods have sought to modify learning [9] or inference [13] so that the final label distribution is likely to *resemble* the expected distribution (e.g., from the training labels) – thus countering the effects of over-propagation. Such "resemblance-based" methods encourage the results towards the expected distribution but are also influenced by the (uncorrected) predicted distribution, so the final distribution does not exactly match the expected distribution. In contrast, some recent work [12] has proposed methods that rank all nodes according to their predictions, then *assign* a label to each node (based on the ranking) so as to produce the expected distribution. These "assignment-based" methods are guaranteed to produce the desired distribution, regardless of the distribution of the predicted labels.

To date, some studies have examined individual methods for correcting relational bias in isolation, but leave a number of important questions unanswered. First, how do the resemblance-based methods compare to assignment-based methods? In particular, does the flexibility of the former approach lead to higher or lower accuracy? Second, the resemblance-based methods can accomplish their goal using modifications to learning or to inference. Are the (computationally simpler) adjustments to inference alone sufficient to yield good accuracy? Third, Gibbs sampling is often considered to be a very accurate inference algorithm [4], yet with LBC it has sometimes produced erratic results and/or accuracy that is inferior to that of "ICA" [14]. Can any of the aforementioned methods for correcting relational bias also improve accuracy with Gibbs? If so, would this alone make Gibbs more competitive with ICA for LBC? Finally, recent work has demonstrated that LBC accuracy can often be substantially improved by the addition of "neighbor attributes" as model features [15]. The addition of such attributes helps to ground inference, since their values typically do *not* need to be estimated (in contrast to the values of neighbor labels). Such grounding should reduce the amount of over-propagation during inference. Thus, does LBC with such features still require explicit methods to reduce relational bias?

Our contributions are as follows. First, we introduce a new assignment-based method, GOALSEEK, and demonstrate how it generalizes existing methods in order to support more than two class labels. Second, we present the first comparison between resemblance-based and assignment-based methods for correcting relational bias. We show that the former typically leads to higher accuracy, and also explain how to add more flexibility to the latter so as to sometimes improves accuracy. Third, we show that among the resemblance-based methods, "CMN" [13] typically yields accuracy that meets or exceeds that of "label regularization" [9]; thus, the simpler, inference-based correction (CMN) may often be preferred. Fourth, we provide the first evaluation of bias correction methods when used with Gibbs sampling. We show here that over-propagation correction methods can yield large accuracy gains, especially but not only when the labels are sparse. In addition, we show for the first time that related relational biases cause Gibbs initialization to have a large impact on accuracy. We then introduce two new methods that, by leveraging more predictive features for initialization, significantly increase accuracy. Overall, we show that with Gibbs, addressing relational bias in two ways (for instance, using appropriate methods for initialization *and* for preventing over-propagation) substantially increases accuracy, leading to performance for Gibbs that is much more competitive with the popular "ICA" method [4].

Finally, we show that correcting for relational bias is, as conjectured above, less essential when neighbor attributes are used. We demonstrate, however, that even in this case these methods continue to be important when the provided labels are very sparse, regardless of whether Gibbs or ICA is used. This shows the importance of adequately addressing relational bias for sparsely-labeled LBC, regardless of the type of collective inference or model features that are used.

II. LINK-BASED CLASSIFICATION (LBC)

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 $Y^K = \{y_i | v_i \in V^K\}$. 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 ($V^U = V \setminus V^K$).

For 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 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.
- **Neighbor labels:** features based on the *labels* of pages that link to v, such as "Count the number of v's neighbors with label `Student`."

Table I characterizes LBC models based on the kinds of features they use. The simplest models use only one kind. For instance, a "content only" or "attribute only" model (ATTRSONLY) uses only self attributes; this is a common baseline. Alternatively, several "relational only" models use only neighbor labels. For instance, wvRN+RL [16] repeatedly averages the predicted label distributions of a node's neighbors; this performs surprisingly well for some datasets.

Typically, the most accurate 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) [3]. CI methods include Gibbs sampling, belief propagation, and ICA (Iterative Classification Algorithm) [4]. Finally, RCI ("relational collective inference") uses all three kinds of features. Because it uses neighbor labels, it also must use some kind of collective inference such as ICA.

Most prior work on LBC has used some form of CI, with collective inference — this is what is typically meant by methods for "collective classification." In this paper, we focus on inference with ICA and Gibbs sampling. ICA is a simple, popular, and effective algorithm [4], [8], [9]. We now describe its use with CI. ICA first performs a "bootstrap" step by predicting a label for every node in V^U using only self attributes. It then constructs additional relational features X_R using the known and predicted node labels (Y^K and Y^U), and re-predicts labels for V^U using both self attributes and X_R. This process of feature computation and prediction is repeated, e.g., until convergence. With RCI, ICA follows the

TABLE I
TYPES OF MODELS, BASED ON THE KINDS OF FEATURES USED.

Model	Self attrs.	Neigh. attrs.	Neigh. labels
ATTRSONLY	✓		
RELATONLY			✓
CI	✓		✓
RI	✓	✓	
RCI	✓	✓	✓

same process, except that self *and* neighbor attributes are used whenever attributes are needed.

An alternative inference method is Gibbs sampling (herein simply GIBBS) (cf., [3]). GIBBS starts with some labeling of the graph (typically, randomly generated), then modifies the graph labeling over a large number of iterations. In each iteration, new probability estimates are computed for each node in the graph (using the current label estimates of neighboring labels). However, instead of using the predictions to select the (new) most likely label for each node, a new label is probabilistically sampled from that node's predicted distribution. Unlike with ICA, GIBBS is not seeking to converge to a single final labeling of the graph. Instead, final label estimates for a node are based on the frequency with which each label was sampled for that node across all iterations.

RI (and RCI) has been used for LBC much less frequently than CI, due in part to early work that showed that RI sometimes hurt accuracy compared to using attributes alone [1] and/or generally underperformed CI [3]. However, a more recent study showed that RI and RCI can lead to significantly higher accuracy than CI — *if* only a small number of labels are available for training [15]. For this reason, we consider CI, RI, and RCI in this paper.

III. APPROACHES TO CORRECTING RELATIONAL BIAS

Section I described the challenges of relational bias, and explained how it leads to two primary problems for LBC: "over-propagation" or "flooding," where large regions of the network are all assigned the same label during inference, and self-feedback loops during SSL, leading to the learning of less accurate models. Both problems tend to lead to situations where the predicted label distribution is not very representative of the true label distribution, and methods that address this class distribution imbalance can address both problems. Therefore, for simplicity (and because most prior work has focused on over-propagation), in the rest of the paper we refer to methods that address both of these problems as simply "over-propagation correction methods."

We divide these methods into two groups: *resemblance-based*, where the method encourages the final results towards some expected distribution but does not (fully) require it, and *assignment-based*, where the final predictions are guaranteed to match some target distribution. Table II summarizes these methods, which are more fully described below.

IV. RESEMBLANCE-BASED BIAS CORRECTION METHODS

A. Label Regularization

Label regularization [17] was designed to make SSL more robust by encouraging a learned logistic regression (LR) classifier (or other exponential family model) to produce probability estimates on the unlabeled data so that the resultant class distribution resembles an expected distribution (which can be estimated from the labeled data). Specifically, it modifies LR's objective function to penalize models to the degree that they produce distributions that are different from what is expected.

Label regularization (LBLREG) was not designed with LBC in mind, but+ McDowell & Aha [9] showed how it can be adapted for LBC, and previously demonstrated that it can significantly increase accuracy. In particular, by encouraging more plausible label distributions during learning, this method helps to address both of the problems with relational bias described in Section I (i.e., the over-propagation that results from collective inference, and the possible self-feedback loops introduced by the use of repeated SSL iterations).

McDowell & Aha showed that LBLREG can significantly increase accuracy, but only evaluated CI. LBLREG is especially important with CI, due to the potential for over-propagation described above. With RCI or RI, however, the potential for over-propagation is somewhat reduced, due to the addition of neighbor attributes to help ground the inference (with RCI) or the elimination of iterative inference (with RI). Thus, is LBLREG still necessary with RCI and RI? Section VIII answers this question.

B. Class Mass Normalization (CMN)

LBLREG is an efficient modification to LR's existing learning process (e.g., using gradient descent), but it does introduce additional computational cost. Could a simpler method achieve comparable results, especially if such methods are less necessary with RCI or RI? In particular, we investigate *class mass normalization* (CMN) [13], which modifies inference, rather than learning. CMN rescales the predicted probabilities of every node so that the total mass of each class matches the expected label distribution (thus making "over-represented" classes in the predicted labels less likely to be selected). Specifically, CMN is a heuristic method that selects the final predicted label for each node as follows

$$y_i = \arg\max_{c \in C} p_i(c) \cdot \frac{\tau_c^*}{\sum_{v_j \in V^U} p_j(c)} \quad (1)$$

where $p_i(c)$ is the predicted probability of node i being label c, τ_c^* is the expected class distribution for class c, and the denominator represents the predicted "mass" of class c.

Zhu et al. [13] show that CMN can increase accuracy, but only used datasets without pre-existing links, and only considered "relational-only" classifiers (i.e., ignoring attributes). McDowell [18] report using CMN in a study of active learning with LBC, but did not evaluate its impact. Section VIII examines whether it can indeed increase LBC accuracy.

TABLE II

SUMMARY OF METHODS FOR PREVENTING THE EFFECTS OF "OVER-PROPAGATION" DUE TO RELATIONAL BIAS, AND DESCRIPTION OF WHERE THESE METHODS HAVE PREVIOUSLY BEEN USED WITH LBC, IF EVER. EACH METHOD ASSUMES THAT WE HAVE AN ESTIMATE OF THE EXPECTED LABEL DISTRIBUTION τ^*, FOR INSTANCE ESTIMATED FROM THE TRAINING DATA.

Method name	Description	Previous LBC use?
	Resemblance-based methods	
Label regularization (LBLREG)	During learning, biases learning process towards models that yield label distributions similar to τ^*.	McDowell & Aha [9], [15]
Class mass normalization (CMN)	During inference, rescales the predicted probabilities of every node so that the total mass of each class matches that of τ^*.	Zhu et al. [13] (but only with "relational-only" classifiers)
	Assignment-based methods	
GOALSEEK	Starting from empty set, repeatedly assign some node v to a class c. At each step, select c so that the distance between the current set's distribution and τ^* is minimized, then choose v (from among the unassigned nodes) with maximal estimated probability for class c.	New to this work. However, produces (for the case of two labels) similar effect to the method of Pfeiffer et al. [12].
GOALSEEKFLEX	Same as GOALSEEK, but uses a different goal distribution τ' to reflect some tendencies of the unlabeled data. $\tau'$is the mean of τ^* and the (unadjusted) distribution predicted by the CI, RI, or RCI model.	New to this work.

C. Challenges with Resemblance-based Methods

LBLREG and CMN both encourage the final distribution to resemble the expected distribution, but do not require it. This may lead to several possible problems. First, since LBLREG affects learning, not inference, it cannot adjust its behavior based on changes that occur to the predicted distribution during inference. In particular, if the iterations of collective inference (e.g., with ICA) begin to drift towards a distribution that LBLREG did not anticipate during learning, it is possible for LBLREG's learned compensations to actually accentuate, rather than minimize, some kinds of over-propagation.

CMN may avoid this problem, since it is applied after every step of collective inference, and thus is continually seeking to "pull" the predicted distribution towards the expected distribution. Its approach, however, is not guaranteed to be successful. Consider one extreme example: suppose there are two class labels A and B and the expected distribution τ^* is $\{A : 60\%, B : 40\%\}$. Suppose further that over-propagation occurs during collective inference, resulting in every node i having predicted probabilities of (approximately) $p_i(A) = 0.9$ and $p_i(B) = 0.1$ (and thus, 100% of the nodes will be classified as class A). Applying CMN will, effectively, rescale the probabilities so that $p'_i(A) = 0.6$ and $p'_i(B) = 0.4$. This satisfies the "total mass" constraints for A and B, but still results in every node being assigned to label A.[1]

The above example is simplified, but demonstrates that CMN's rescaling of probabilities to match a "total mass" constraint does not necessarily result in a set of label predictions that resembles the expected distribution τ^* (and indeed may not change any label predictions). Thus, neither LBLREG nor CMN is guaranteed to result in a label distribution that is very similar to τ^*. Do they nonetheless work well in practice? Alternatively, would *assignment-based* methods that guarantee that the final distribution matches τ^* yield better results? The next section introduces the assignment-based methods, while Section VIII answers these questions empirically.

[1]An alternative approach to this challenge would be to use a"soft" method for collective inference (i.e., one that uses the full probability distribution for each node, rather than simply using the most likely predicted label). Such methods have been used much less frequently for LBC, due to their greater complexity and the lack of evidence that they improve upon "hard" labeling methods such as ICA and Gibbs. However, some recent studies [19], [12] have used soft methods, and future work should compare against this approach.

Fig. 1. GOALSEEK & GOALSEEKFLEX algorithm; for notation, see Sect. II.

```
GoalSeek ($V^U, \tau^*, p, mode$)=
// $V^U$=nodes with unknown labels, $\tau^*$=expected class distribution
// $p$=predicted label probabilities for $V^U$,$mode$=algorithm type
1   $\tau \leftarrow \emptyset$              // Initially, current distribution $\tau$ is empty
2   $\mathcal{U} \leftarrow V^U$             // and all nodes in $V^U$ are unassigned
3   $\tau' \leftarrow \tau^*$                // Use expected distribution $\tau^*$ as goal...
4   if $mode = Flex$   // or, adjust goal based on predictions $p$
5       $\tau' \leftarrow average(\tau^*, computeLabelDistribution(p))$
6   while $\mathcal{U} \neq \emptyset$       // Repeat until no unassigned nodes remain
7       $c \leftarrow mostNeededClass(\tau, \tau')$ // Find best class to add to $\tau$
8       $i \leftarrow \arg\max_{j|v_j \in \mathcal{U}} p_j(c)$  // In $\mathcal{U}$, find most likely node for $c$
9       $y_i \leftarrow c$                   // Assign node $v_i$ to class $c$
10      $\mathcal{U} \leftarrow \mathcal{U} \setminus \{v_i\}$   // Remove $v_i$ from unassigned set
11      $\tau \leftarrow \tau \cup \{c\}$     // Update current distribution $\tau$
12  return $\{y_i|v_i \in V^U\}$  // Return nodes' label assignments
```

V. ASSIGNMENT-BASED BIAS CORRECTION METHODS

The previous section described several methods for correcting relational bias that, while likely useful, are not guaranteed to produce very plausible label distributions. In response, this section presents two new methods for correcting relational bias. Both methods directly *assign* labels to each node so as to guarantee a final label distribution that matches some desired distribution, while adhering as much as possible to the probabilities p predicted by the unadjusted LBC model.

The first method, GOALSEEK, operates as follows (see Figure 1). Steps 1-2 perform initialization, and in particular add all nodes in V^U (those with "unknown" labels, see Section II) to the set of "unassigned" nodes $\mathcal{U}$. Steps 3-5 select the goal distribution τ', as discussed further below. Each iteration of the loop (steps 6-11) will then assign some node v_i to a specific class c. In particular, step 7 selects the class c whose addition to the current distribution τ would make τ most similar to the goal τ' (e.g., by minimizing the Euclidean distance). Next, step 8 identifies the unassigned node v_i that is most likely to have true class c (based on the probabilities p). Steps 9-11 then explicitly assign v_i to class c and update τ and $\mathcal{U}$. This process repeats until all nodes have been assigned labels.

As its goal, GOALSEEK seeks to make the final label distribution as close as possible to the expected distribution τ^*, which can be estimated from the training data (e.g., based on the known labels Y^K). However, the true distribution of Y^U, the labels for the unknown nodes, will likely differ to some extent from Y^K, especially if $|Y^K|$ is small. For this reason, we propose a new method: GOALSEEKFLEX is identical to GOALSEEK except that in Step 5 of Figure 1 it sets the goal distribution τ' equal to the average of the expected distribution τ^* and the distribution computed from the raw label predictions p. In theory, including the *expected* distribution τ^* should help to restrict over-propagation, while using the *predicted* probabilities (as computed for the unknown nodes) should help account for differences between τ^* and the true distribution of Y^U. Section VIII evaluates how well this succeeds in practice.

Recent work by Pfeiffer et al. [12], in the context of "soft" inference methods (see Footnote 1), proposed an assignment-based method that is closely related to GOALSEEK. Their method first ranks all nodes by their predictions, then shifts the probabilities so as to partition the ranked nodes in a way that produces exactly the desired distribution. Their method only handle datasets with two class labels; in this case it produces exactly the same distribution as that of GOALSEEK, when used with a "hard" inference method like ICA or GIBBS. In contrast, our methods handle any number of class labels. Also, they show that their method improves accuracy compared to not using any bias adjustment, but do not compare against competing methods such as CMN or LBLREG.

VI. ADDRESSING RELATIONAL BIAS WHILE USING GIBBS

Section II described two common collective inference methods, ICA and GIBBS. ICA has been more frequently used with LBC, due to a combination of its simplicity and its typically strong relative accuracy. In contrast, performance comparisons have sometime found results with GIBBS to be erratic [14]. On the other hand, GIBBS is generally considered (in non-LBC contexts) to be a very accurate inference algorithm [4]. Could addressing relational bias improve its LBC accuracy? If so, would GIBBS accuracy approach or even exceed that of ICA?

This section considers two different manifestations of relational bias as it applies to GIBBS, and considers how to address both cases. First, we consider how to prevent/address over-propagation, by adapting some methods that were described above. Second, we consider modifying the GIBBS initialization, to prevent bias in the later iterations from trapping GIBBS in a non-representative "island" of the probability space.

A. Addressing Bias during Sampling

The methods listed in Table II all help to prevent over-propagation. However, not all are applicable to GIBBS. In particular, at each iteration, GIBBS must probabilistically sample a label for each node, based on its current label predictions. Thus, methods that *assign* a label without modifying the probabilities, such as GOALSEEK and GOALSEEKFLEX, are not sensible to use with GIBBS. On the other hand, LBLREG integrates easily with GIBBS, since it modifies learning in a way that will directly impact the estimated label probabilities.

Using CMN with GIBBS is possible but requires minor modification. Instead of rescaling the probabilities and then selecting the class c with the highest resultant score (as in all previous work with CMN, and as reflected by the $argmax$ in Equation 1), we simply adapt CMN's idea of rescaling to create new "probabilities" as follows:

$$p'_i(c) = \alpha \cdot p_i(c) \cdot \frac{\tau^*_c}{\sum_{v_j \in V^U} p_j(c)} \tag{2}$$

where α is a normalizing constant used to ensure that $\sum_{c \in C} p'_i(c) = 1$. GIBBS (re)computes these values $p'(c)$ at each iteration, then uses them for its probabilistic sampling.

This approach, with CMN, helps to prevent over-propagation by tending to reduce the number of nodes that are sampled into "over-represented" classes, and thus should be expected to improve accuracy. On the other hand, to produce accurate inference results, GIBBS requires accurate probabilities — and although Equation 2's correcting for bias may, on average, produce more reasonable probabilities, this use of the CMN heuristic for probability modification may not always yield accurate, well-calibrated probabilities. Thus, we may expect LBLREG's more principled modification of the objective used during learning to produce better results for at least some cases. The relative merit of both techniques is unknown, since neither LBLREG nor (modified) CMN has previously been evaluated with GIBBS. Section VIII examines their impact, and whether our conjecture regarding the likely behavior of LBLREG vs. CMN for GIBBS is true.

B. The Challenge of Gibbs Initialization

As described in Section II, GIBBS starts with some initial labeling of the graph. Typically, some form of random selection is used to assign a class label $c \in C$ to each node. This random labeling is unlikely to be very close to a likely labeling of the graph. For that reason, GIBBS implementations typically discard the first N_B iterations after initialization, instead recording statistics only after the "burn-in" period (of perhaps $N_B = 100$ iterations) has elapsed, after which the algorithm has hopefully transitioned to a more representative labeling. In addition, GIBBS implementations will sometimes execute multiple "chains" of inference, where each chain begins from a new (initialized) labeling, and executes N_B burn-in iterations before statistics are recorded. This helps GIBBS to explore more of the probability space, and helps the sampling to escape from regions of the probability space that are not well-connected to the rest of the space.

Even with the use of multiple chains and a substantial burn-in period, GIBBS may be susceptible to getting stuck in a region of the probability space that does not represent a very likely overall labeling. In particular, GIBBS is well-known to struggle when provided with extreme conditional probabilities, a situation that is very likely to occur with LBC when significant auto-correlation (i.e., highly predictive links) [3] is present. This form of relational bias directly interacts with the initial node labeling. For example, if some regions of the graph start with a few interconnected nodes that (randomly) happen to have the same label, the learned relational dependencies may lead to a set of extreme, self-reinforcing probabilities that creates a fairly-isolated "island" in the probability space that GIBBS is unlikely to escape from, even if the corresponding labeling is not very likely to be true. Thus, starting with a more plausible labeling may be important to improving overall accuracy.

We consider five different initialization methods in this paper. Three of these have been used by prior work:

1) RANDUNIINIT: Labels selected *uniformly* at random from the label set C.
2) RANDPROPINIT: Labels probabilistically sampled from (i.e., in *proportion* to) the expected distribution τ^*.
3) SELFATTRINIT: For each node, select the most likely label predicted by a learned "self attribute only" model.

We also propose and study two new initialization methods:

4) BOTHATTRINIT: Like SELFATTRINIT, but uses a model with features based on *both* self and neighbor attributes. Because neighbor attributes can be highly informative, this may lead to a much more accurate initial labeling.
5) ONESTEPINIT: Predict a label for each unknown node with a "bootstrap" model that uses only attributes (see Section II), then use the predicted labels to perform *one step* of ICA to generate the initial labels for GIBBS. ICA's single step uses the same model that is later used with each GIBBS iteration. Thus, with CI it uses self attributes and neighbor labels, and with RCI it also uses neighbor attributes (GIBBS does not apply to RI).

Observe that the list above is ordered based on increasing amounts of information used for the initialization, starting from no information (RANDUNIINIT), to using basic knowledge of the class distribution (RANDPROPINIT), to using various amounts of attribute and label information (SELFATTRINIT, BOTHATTRINIT, and ONESTEPINIT). Note also that the last three methods are entirely deterministic. Thus, with these methods each of the separate GIBBS "chains" will begin with exactly the same labeling. However, GIBBS' probabilistic sampling will cause each of those chains to immediately diverge, so there is still value in using separate chains.

Prior work with GIBBS for LBC has used, for instance, RANDUNIINIT [20], [21], RANDPROPINIT [5], and SELFATTRINIT [4], [14], though frequently the method is not fully specified, or the initialization is not described at all (e.g., [22], [10]). We are not aware of any prior work that has used BOTHATTRINIT or ONESTEPINIT.[2]

[2]A possible exception is that of Sen et al. [4]. They use CI, and appear to perform initialization using self attributes and neighbor labels, and thus are similar to ONESTEPINIT. Crucially, however, they did not include the "bootstrap" step for the initialization. Thus, prediction will be able to use only the (very small number of) known labels, and thus in practice will depend only on the self attributes (i.e., equivalent to SELFATTRINIT).

TABLE III
DATA SETS SUMMARY.

Characteristics	**Cora**	**CiteSeer**	**Gene**
Total nodes	2708	3312	1103
Total links	5278	4536	1672
Class labels	7	6	2
% dominant class	16%	21%	56%

In addition, no prior work has directly studied the impact of GIBBS initialization for LBC. One partial exception is that of Neville & Jensen [20] who examined LBC with fully-labeled training graphs, and observed poor results with GIBBS, using RANDUNIINIT. They reported, informally, that a non-random initial labeling could improve accuracy, but did not provide details on the method or the results. We study this problem in the more challenging case of a partially-labeled training graph, and specifically evaluate what kind of initialization is most helpful and necessary for overcoming the relational bias that was described above.

VII. EXPERIMENTAL METHOD

A. Datasets and Features

We used three commonly-used real datasets (see Table III). **Cora** (see [4]) is a collection of machine learning papers. **Citeseer** (see [4]) is a collection of research papers drawn from the Citeseer collection. For both datasets, attributes represent the presence of particular words, and citations provide links between the documents. We ignored link direction, as with Bilgic et al. [8]. They also report substantially higher accuracies using principal component analysis (PCA) to reduce the dimensionality of the attributes, so to provide a stronger baseline we mimic their setup and use the 100 top attribute features after applying PCA to the entire graph.

Gene (see [3]) describes the yeast genome at the protein level; links represent protein interactions. We mimic Xiang & Neville [7] and predict protein localization using four attributes: Phenotype, Class, Essential, and Chromosome. We binarized these attributes, yielding 54 binary attributes.

B. Classifiers and Regularization

CI, RI, and RCI all require learning a classifier to predict the label based on self attributes, and (for RI and RCI) a separate classifier based on neighbor attributes. We use logistic regression (LR), because it usually outperformed other alternatives [4], [8], [15]. To combine the predictions from a varying of number of neighbors' attributes, RI and RCI use the "MNAC" method [15].

RCI and CI also require a classifier to predict a node's label based on its neighbors' labels. McDowell & Aha [9] found that Naive Bayes (NB) with "multiset" features was superior to LR with "proportion" features as used by Bilgic et al. [8]. Thus, we use NB for neighbor labels, and combine these results with the LR classifiers used for other features (described above), using the "hybrid model" method [9]. This approach applies the classifiers separately, using different sets of features, then combines them with a probabilistic formula.

TABLE IV

Results with ICA. AVERAGE ACCURACIES FOR VARYING VALUES OF THE LABEL DENSITY d, USING ICA FOR COLLECTIVE INFERENCE. WITHIN EACH SECTION (FOR CI, RI, AND RCI) WE BOLD THE BEST VALUE AND ALL VALUES THAT WERE *not* STATISTICALLY DIFFERENT FROM THAT VALUE. NOTE THAT HIGHER VARIANCE CAUSES SOME APPARENTLY SMALLER DIFFERENCES TO NOT BE STATISTICALLY SIGNIFICANT. THE ROW FOR NOADJUST SHOWS THE BASELINE RESULTS WHERE NO METHOD FOR OVER-PROPAGATION CORRECTION IS APPLIED.

	Cora				Citeseer				Gene			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	1%	3%	5%	10%
CI-NOADJUST	45.5	75.1	78.7	82.1	47.9	64.2	**66.9**	**69.9**	59.9	64.3	**70.9**	**76.5**
CI-LBLREG	62.9	**78.4**	**80.2**	**82.1**	**57.6**	**65.9**	**67.8**	**69.6**	**62.8**	66.4	**73.4**	**78.7**
CI-CMN	60.1	**77.5**	**80.0**	**82.5**	53.7	**65.3**	**68.0**	**69.9**	**63.8**	**67.7**	**72.0**	**78.5**
CI-GOALSEEK	**66.7**	76.3	78.9	80.0	**58.7**	**65.9**	**67.4**	69.3	**63.0**	**69.9**	**74.6**	77.2
CI-GOALSEEKFLEX	**68.0**	76.5	79.2	80.5	**59.2**	**65.8**	**67.6**	**69.5**	**63.3**	**69.8**	**73.6**	**78.1**
RI-NOADJUST	67.8	**78.7**	**80.0**	**81.3**	**61.7**	**69.3**	**70.5**	**71.6**	64.5	71.5	**75.9**	**78.4**
RI-LBLREG	**71.3**	**78.9**	**80.3**	**81.5**	**62.8**	68.3	69.6	71.2	**68.3**	**72.8**	**76.0**	**77.6**
RI-CMN	**72.0**	**79.4**	**80.4**	**81.6**	**62.1**	**69.1**	**70.3**	**71.4**	**67.1**	**74.1**	**75.4**	**78.2**
RI-GOALSEEK	68.6	76.8	78.4	79.0	61.4	67.1	68.2	70.0	**68.2**	**72.4**	**75.6**	**77.7**
RI-GOALSEEKFLEX	69.4	77.1	78.3	79.7	**62.0**	67.6	68.8	70.5	**68.0**	**72.7**	**75.2**	**77.6**
RCI-NOADJUST	70.6	**79.6**	**80.9**	**82.4**	**61.6**	**68.9**	**70.3**	**71.8**	67.4	**75.3**	**77.1**	**79.5**
RCI-LBLREG	**72.3**	**79.7**	**81.2**	**82.6**	**63.3**	**68.4**	69.6	71.2	**70.1**	**75.6**	**77.5**	**79.1**
RCI-CMN	**73.6**	**80.2**	**81.1**	**82.6**	62.3	**68.7**	**70.3**	71.5	**69.7**	**75.9**	**77.3**	**79.4**
RCI-GOALSEEK	69.2	77.7	79.6	80.5	60.9	66.9	68.3	70.3	**70.1**	**75.2**	**77.5**	**78.8**
RCI-GOALSEEKFLEX	70.0	78.2	79.7	81.2	60.9	67.2	68.4	70.3	**70.1**	**75.4**	**77.0**	**79.1**

For sparsely-labeled data, regularization can have a large impact on accuracy. To ensure fair comparisons, we used five-fold cross-validation, selecting the value of the regularization hyperparameter that maximized accuracy on the held-out labeled data. We used a Gaussian prior with all LR's features and a Dirichlet prior with NB's discrete features (for neighbor labels).

C. Learning and Collective Inference

For inference and semi-supervised learning, we chose methods that performed well in prior comparisons [4], [8], [9]. RI does not require iterative collective inference; CI and RCI used ICA or GIBBS. For ICA, we used 10 iterations (see Section II), a common choice that is generally sufficient for (approximate) convergence. For GIBBS, we used 1500 iterations, with a random restart every 300 iterations, and ignored the first 100 iterations after a restart for burn-in (i.e., a total of 5 chains); this is comparable to the number of iterations used with these datasets in prior work [5], [20], [14].

For learning, we use the "SSL-EM" approach [7], [9], [12]: repeatedly predict the labels (with CI, RI, or RCI) for all unknown nodes, then relearn models using predicted and known labels. We use 10 such SSL iterations.

D. Correcting for Over-propagation

We consider all of the methods from Table II, in addition to NOADJUST (LBC performed without any correction for over-propagation). The methods of Table II all require some estimate of the expected distribution τ^*; we compute this using the known labels Y^K, with Laplace smoothing.

E. Evaluation Procedure

We report accuracy averaged over 20 trials. For each trial, we randomly select some fraction (the "label density" d) of V to be "known" nodes V^K; those remaining form the "unknown label" test set V^U. We focus on the important sparsely-labeled case [6], [9], e.g., $d \leq 10\%$. Accuracy is evaluated only over the unknown labels.

In the results below, all uses of the term "significant" refer to statistical significance conclusions based on one-tailed paired t-tests accepted at the 95% confidence level.

VIII. RESULTS

We first examine results when ICA is used for inference, then consider results with Gibbs.

A. Results with ICA

Table IV shows the average accuracy, when using ICA, of the four different methods for preventing over-propagation, compared to using no such adjustment (NOADJUST). We vary the label density d from 1% to 10%. Within each section (for CI, RI, and RCI), we bold the best accuracy and all other values that were not significantly *worse* than that value.

Result: With CI, all of the over-propagation correction methods substantially improve accuracy when labels are sparse. Bias correction is most important when there are fewer known labels to guide learning and to ground ICA's inference. Thus, when d is small, correcting for over-propagation leads to large gains compared to no correction, e.g., with LBLREG significant gains of up to 17% for Cora, 10% for Citeseer, and 3% for Gene. As d increases, these gains (for most of the correction methods) diminish but correction remains helpful for all $d \leq 5\%$ for Cora and Citeseer and for all values of d shown for Gene.

Result: When neighbor attributes are used (with RI or RCI), the need for over-propagation correction decreases but remains important when the labels are sparse. We had conjectured that bias correction in general would be less important when using models that include neighbor attributes, but no prior work had actually evaluated this claim. Table IV shows that our conjecture was correct: in 10 of the 12 cases for

RCI and 9 of the 12 cases for RI, NoAdjust is statistically equivalent to the top performing over-propagation correction method. However, correction continues to be helpful when the labels are very sparse. For instance, with RI or RCI, using CMN increases accuracy in all cases for $d \leq 3\%$ (except for $d = 3\%$ with Citeseer), by a range of 0.4-4.2% for RI and by 0.6-3.0% for RCI. Thus, when neighbor attributes are used, including over-propagation correction is less essential, but still often helpful if the labels are sparse.

Result: Overall, the "resemblance-based" methods typically yield higher accuracy than the more rigid "assignment-based" methods. The assignment-based methods (GoalSeek and GoalSeekFlex) very rarely outperform the resemblance-based methods (LblReg and CMN); one exception is for CI with $d = 1\%$ (discussed later). Instead, CMN and LblReg almost always have significantly better accuracy than the GoalSeek variants for Cora and Citeseer, and often have better accuracy with Gene (though the amounts are generally small and not significant).

In general, CMN and LblReg outperform NoAdjust because the relational bias discussed in Section I can lead to cascading errors, when the labels are sparse. Thus, using some adjustment method, to ensure that the predictions do not deviate too far from the expected distribution, is helpful. However, the correction approach of CMN and LblReg tends to yield better results than GoalSeek because GoalSeek enforces the goal of matching the expected distribution too strictly, without leaving any room for flexibility based on the difference between the expected distribution (from the training data) and the true distribution of the unknown labels. Notably, GoalSeekFlex almost always performs somewhat better than GoalSeek (GoalSeek has better accuracy in only 8 of the 36 cases shown, and often by very small amounts), showing that the *flexibility* that GoalSeekFlex provides (by adjusting the target distribution to partially reflect the predicted distribution) is helpful. Nonetheless, the even greater flexibility of CMN and LblReg leads to better accuracy than with GoalSeekFlex (except for a few cases for CI at $d = 1\%$ where the GoalSeek variants perform especially well).

Discussion: Overall, the two resemblance-based methods, CMN and LblReg, usually yielded the best accuracy and produced results that were fairly similar to one another. LblReg had a tendency to produce somewhat better accuracy for $d = 1\%$; these gains were significant in 3 of 9 cases. As a whole, however, CMN produced results that were very comparable to that of LblReg, yet CMN has the advantage of being computationally much simpler (since it is a trivial modification of probabilities during inference, rather than an adjustment to the learning objective). Thus, future researchers should consider using CMN as a baseline for future studies of bias correction, and routinely include CMN in other studies in order to prevent such bias from leading to degenerate results.

B. Results with Gibbs Sampling

Table V shows the overall results when different variants of bias correction are applied to Gibbs sampling. Results are shown for CI and RCI (Gibbs doesn't apply to RI). Within each vertical section (for CI and RCI), the first row shows the baseline results as typically used by prior studies (where initialization is done randomly, and no over-propagation correction method is used). The second row shows the result of adding the strongest reasonable initialization method (OneStepInit, as discussed later). Finally, the third and fourth rows of each section show the results of combining OneStepInit with CMN and LblReg, respectively.

Result: combining a strong initialization method with an over-propagation correction method produces significant, and often substantial, gains in Gibbs accuracy. In Table V, comparing the first row of each section (baseline Gibbs as used in most prior studies) with the third or fourth row shows substantial gains. For instance, with CI comparing the baseline to OneStepInit-LblReg shows gains of 5-23% for Cora, 12-17% for Citeseer, and 4-7% for Gene. With RCI, these gains are reduced, but are still significant (in all but one case), and range from 5-7% for Citeseer and 2-4% for Gene. Closer examination of the table reveals that the gains generally come from over-propagation correction (e.g., CMN or LblReg) as well as from using OneStepInit. As expected, the gains are generally largest when d is small, though some of the largest gains with Gene occur when d is as high as 10%.

Result: for Gibbs, CMN and LblReg both perform well with RCI, but LblReg is often much better with CI. With CI, CMN always improves accuracy compared to NoAdjust, but LblReg often produces much better accuracy (up to 6% better), especially with Cora and Citeseer, when the labels are sparse. CMN yields better accuracy in only 3 of the 12 cases shown, and always by less than 1%. In contrast, with RCI the addition of neighbor attributes helps to limit over-propagation. Thus, using some method for over-propagation correction still improves accuracy compared to NoAdjust, when labels are sparse, but the differences are much smaller and there is no clear winner between CMN and LblReg.

Result: the type of Gibbs initialization has a large effect on the overall accuracy, and using the two new methods consistently leads to better results. Table VI again shows the results of inference with Gibbs, but where we vary the initialization method. We use LblReg as the over-propagation correction method, due to its consistently strong performance in Table V, but the relevant trends with NoAdjust or with CMN were similar. The results show that using a non-random initialization consistently and significantly outperforms the random initialization methods, for every case with CI and for almost every case with RCI. Even comparing against the maximum of the two random methods, initialization with OneStepInit often yields gains of more than 20% for Cora and Citeseer with CI, and gains of 3-10% Gene. These gains are smaller with RCI, but still significant and reach as much as 3% for Cora, 10% for Citeseer, and 4% for Gene.

We had expected that initialization would improve Gibbs accuracy, but that simple methods such as SelfAttrInit would suffice for providing a reasonable starting point, so that

TABLE V

Overall Results with GIBBS. AVERAGE ACCURACIES FOR VARYING VALUES OF THE LABEL DENSITY d, USING GIBBS FOR COLLECTIVE INFERENCE. WITHIN EACH SECTION (FOR CI, RI, AND RCI) WE BOLD THE BEST VALUE AND ALL VALUES THAT WERE *not* STATISTICALLY DIFFERENT FROM THAT VALUE. THE FIRST ROW OF EACH SECTION (RANDPROPINIT-NOADJUST) REPRESENTS THE TYPICAL BASELINE ACCURACY ACHIEVED WITHOUT THE IMPROVEMENTS FOR BIAS CORRECTION DESCRIBED IN THIS PAPER.

	Cora				Citeseer				Gene			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	1%	3%	5%	10%
CI-RANDPROPINIT-NOADJUST	36.5	60.6	65.9	77.4	39.5	49.7	51.7	56.3	58.5	61.4	65.8	71.0
CI- ONESTEPINIT -NOADJUST	44.3	73.7	77.6	**81.5**	43.1	55.6	58.1	64.8	59.7	59.2	60.9	67.8
CI- ONESTEPINIT -CMN	**53.1**	75.9	78.7	**81.9**	51.1	**63.6**	**66.4**	**69.0**	**63.6**	**66.4**	69.8	**76.9**
CI- ONESTEPINIT -LBLREG	**59.8**	**78.3**	**80.2**	**82.0**	**56.2**	**64.8**	**66.6**	68.5	**62.2**	**65.1**	**73.3**	**77.7**
RCI-RANDPROPINIT-NOADJUST	71.0	78.2	79.9	81.9	57.2	62.7	63.9	64.3	**67.7**	71.0	73.2	76.4
RCI- ONESTEPINIT -NOADJUST	70.6	**80.0**	**81.1**	**82.7**	60.6	**68.5**	**69.9**	**71.5**	67.2	**74.9**	**77.5**	**79.9**
RCI- ONESTEPINIT -CMN	**73.3**	**80.2**	**81.1**	**82.8**	61.7	**68.6**	**70.0**	**71.5**	**69.6**	**76.3**	**77.9**	**79.7**
RCI- ONESTEPINIT -LBLREG	**72.6**	**79.9**	**81.3**	**82.8**	**62.9**	68.0	**69.5**	**71.2**	**70.2**	**75.4**	**77.6**	**79.1**

TABLE VI

Results with GIBBS, using LBLREG **but varying the initialization method**. AVERAGE ACCURACIES FOR VARYING VALUES OF THE LABEL DENSITY d, USING GIBBS FOR COLLECTIVE INFERENCE AND LBLREG. WITHIN EACH SECTION (FOR CI, RI, AND RCI) WE BOLD THE BEST VALUE AND ALL VALUES THAT WERE *not* STATISTICALLY DIFFERENT FROM THAT VALUE. FOR CI, BOTHATTRINIT IS SHOWN FOR REFERENCE BUT EXCLUDED FROM THIS STATISTICAL COMPARISON BECAUSE THE USE OF NEIGHBOR ATTRIBUTES IS NOT NATURALLY SUPPORTED BY A CI MODEL.

	Cora				Citeseer				Gene			
Label Density (d)	1%	3%	5%	10%	1%	3%	5%	10%	1%	3%	5%	10%
CI- RANDUNIINIT -LBLREG	21.4	26.5	47.3	64.7	28.8	34.9	39.3	50.2	59.0	60.6	64.4	67.7
CI-RANDPROPINIT-LBLREG	32.9	36.5	45.9	66.7	29.6	37.2	37.0	47.4	57.8	59.1	64.5	67.5
CI- SELFATTRINIT -LBLREG	44.4	67.3	74.5	77.8	41.0	46.2	47.4	54.9	**61.7**	**63.0**	69.0	74.3
CI- ONESTEPINIT -LBLREG	**59.8**	**78.3**	**80.2**	**82.0**	**56.2**	**64.8**	**66.6**	**68.5**	**62.2**	**65.1**	**73.3**	**77.7**
CI-BOTHATTRINIT-LBLREG	71.2	79.4	80.9	82.5	59.3	66.9	68.6	70.4	64.5	72.0	78.3	79.7
RCI- RANDUNIINIT -LBLREG	**67.3**	75.9	77.3	79.8	55.5	57.7	58.1	60.1	**69.7**	72.4	73.6	75.8
RCI-RANDPROPINIT-LBLREG	**68.3**	77.5	78.5	81.5	55.4	58.8	59.6	61.8	68.0	71.4	73.4	76.5
RCI- SELFATTRINIT -LBLREG	**66.8**	78.3	79.9	**82.2**	53.9	57.0	57.5	60.2	68.0	70.6	73.3	77.6
RCI-BOTHATTRINIT-LBLREG	**73.1**	79.4	80.9	82.3	**63.0**	**67.7**	69.3	**71.1**	**70.4**	**75.8**	**77.6**	**79.2**
RCI- ONESTEPINIT -LBLREG	**72.6**	**79.9**	**81.3**	**82.8**	**62.9**	**68.0**	**69.5**	**71.2**	**70.2**	**75.4**	**77.6**	**79.1**

the new methods (BOTHATTRINIT and ONESTEPINIT), which utilize additional information during initialization, might provide minimal additional benefit. However, Table VI shows otherwise: the two new methods consistently improve accuracy, even with RCI and when d is as high as 10%. This suggests that Gibbs is heavily influenced by the relational bias caused by the strong auto-correlation in these datasets, despite our use of multiple Gibbs chains with random restarts (see section VI). Interestingly, with CI, the use of neighbor attributes during initialization, with BOTHATTRINIT, leads to substantial gains compared to using ONESTEPINIT — even though, for CI, neighbor attributes are *not* used during the Gibbs iterations (after initialization completes). This further reinforces the importance of good initialization when using Gibbs, and suggests that using a more complex predictive model for initialization may be worthwhile, even if that more complex model would be too computationally expensive to using during the subsequent thousands of Gibbs iterations.

Discussion: Inference with ICA **vs. with** GIBBS**.** If no corrections for relational bias are made, then ICA generally yields substantially better accuracy than GIBBS. For instance, comparing ICA with NOADJUST (see Table IV) vs. the corresponding rows for Gibbs with RANDPROPINIT-NOADJUST (see Table V) shows gains for ICA, with Citeseer, of 8-15% with CI and 6-12% with RCI. However, if LBLREG is used to prevent over-propagation, and GIBBS uses ONESTEPINIT, then the relative performance of GIBBS compared to ICA improves remarkably. For instance, in that case, with Citeseer, ICA beats GIBBS by no more than 1.2% with CI and 0.4% with RCI. Indeed, when using RCI with LBLREG and ONESTEPINIT, GIBBS' accuracy is almost *indistinguishable* from ICA's on all three datasets (accuracies differ by no more than 0.4%); similar results are true if CMN is used instead.

Thus, if suitable methods for correcting relational bias are used, then, at least with RCI, GIBBS' accuracy may be on par with that of ICA. Analysts may still prefer ICA, since it needs only about 10 iterations instead of the thousands needed for GIBBS. Note though that these iterations can be fairly efficient, since most of the computation for each node's predictions can be cached (e.g., for all features based on the unchanging attributes, rather than the predicted labels). Moreover, ICA seeks to converge to a single most likely global assignment of labels to the nodes (i.e., MAP inference), whereas GIBBS is essentially computing the marginal probabilities of each node separately. Some analysis tasks are more suited to marginal inference, in which case employing GIBBS, with the bias corrections described in this paper, could be very helpful.

IX. CONCLUSION

Link-based classification (LBC) can substantially increase accuracy in a range of contexts. Given a dataset with highly predictive links, LBC combined with appropriate semi-supervised learning (SSL) can be highly effective even when given only a sparsely-labeled network for learning and inference. However, the interplay between highly predictive links, strong SSL, and few known labels can sometimes produce degenerate behavior. This behavior can manifest itself in several ways, including collective inference converging to situations where large regions of the network all have the same class label, and/or in cases where a Gibbs sampler remains stuck in a undesirable, "non-representative" state labeling.

These problems result from different manifestations of relational bias. Prior work has sought to address some of these problems, but as described in Section I had a number of limitations. Notably, previous work had not compared against competing methods for preventing over-propagation, had often not evaluated the methods against "no adjustment" baselines, and had not considered how to handle similar problems with relational bias that afflict inference with Gibbs sampling. Moreover, no prior work had examined these challenges in light of the recent addition of neighbor attributes as LBC features — does this addition change the necessity or effectiveness of bias correction?

In response, we provided the first comparison between resemblance-based and assignment-based methods for preventing over-propagation. We showed that the more flexible variety (resemblance-based methods) typically provides the best accuracy while, within the assignment-based methods, providing more flexibility to reflect the observed but unlabeled network (with GOALSEEKFLEX) generally increases accuracy. We demonstrated that correcting for over-propagation can provide large accuracy gains, regardless of what form of collective inference (i.e., ICA or GIBBS) is used. However, we further showed that, with GIBBS, it is also important to address a second manifestation of relational bias, by employing a more accurate method for initialization. Indeed, we found that more accurate initialization consistently increased accuracy, and that one of our new methods (ONESTEPINIT) consistently yielded the best accuracy. Overall, combining a strong initialization method (ONESTEPINIT) with a strong over-propagation correction method greatly improved the accuracy of GIBBS, making it much more competitive with ICA when RCI was used. This makes GIBBS much more feasible for use in cases where inference for marginal probabilities is preferred.

We also provided the first insight into the relative benefits of the two resemblance-based methods, CMN and LBLREG. We found that, when using ICA, both performed very well, and thus the computationally simpler method (CMN) is likely to be preferred. In contrast, we showed that, with GIBBS, LBLREG often yielded better accuracy, at least when CI (rather than RCI) was used. This was consistent with our argument that LBLREG provided a more principled adjustment of the probabilities (by modifying learning), as opposed to CMN's heuristic rescaling of inferred probabilities.

Our results emphasize the importance for LBC of correcting for relational bias, regardless of the type of inference or model features that are used, and provide guidance for the most effective bias correction methods to use in different contexts. These results should be confirmed with additional learning algorithms, inference algorithms, and datasets. Also, many methods that we discussed relied upon some definition of the "expected distribution" of the unknown labels, which we estimated from the known labels. Future work should consider whether the relative performance of the relational bias correction methods changes if a more precise estimate of this distribution were available.

ACKNOWLEDGMENTS

Thanks to the anonymous referees for comments that helped to improve this work. This work was supported in part by NSF award number 1116439 and a grant from ONR.

REFERENCES

[1] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," in *Proc. of SIGMOD*, 1998, pp. 307–318.

[2] J. Neville and D. Jensen, "Iterative classification in relational data," in *Proc. of the Workshop on Learning Statistical Models from Relational Data at AAAI-2000*, 2000, pp. 13–20.

[3] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *Proc. of KDD*, 2004, pp. 593–598.

[4] P. Sen, G. Namata, M. Bilgic, L. Getoor, B. Gallagher, and T. Eliassi-Rad, "Collective classification in network data," *AI Magazine*, vol. 29, no. 3, pp. 93–106, 2008.

[5] J. Neville and D. Jensen, "Relational dependency networks," *Journal of Machine Learning Research*, vol. 8, pp. 653–692, 2007.

[6] X. Shi, Y. Li, and P. Yu, "Collective prediction with latent graphs," in *Proc. of CIKM*, 2011, pp. 1127–1136.

[7] R. Xiang and J. Neville, "Pseudolikelihood EM for within-network relational learning," in *Proc. of ICDM*, 2008, pp. 1103–1108.

[8] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," in *Proc. of ICML*, 2010, pp. 79–86.

[9] L. K. McDowell and D. W. Aha, "Semi-supervised collective classification via hybrid label regularization," in *Proc. of ICML*, 2012, pp. 975–982.

[10] R. Xiang and J. Neville, "Understanding propagation error and its effect on collective classification," in *Proc. of ICDM*, 2011, pp. 834–843.

[11] M. Bilgic and L. Getoor, "Effective label acquisition for collective classification," in *Proc. of KDD*, 2008, pp. 43–51.

[12] J. J. Pfeiffer III, J. Neville, and P. N. Bennett, "Overcoming relational learning biases to accurately predict preferences in large scale networks," in *Proc. of WWW*, 2015, pp. 853–863.

[13] X. Zhu, Z. Ghahramani, and J. D. Lafferty, "Semi-supervised learning using gaussian fields and harmonic functions," in *Proc. of ICML*, 2003, pp. 912–919.

[14] L. K. McDowell, K. M. Gupta, and D. W. Aha, "Cautious collective classification," *Journal of Machine Learning Research*, vol. 10, pp. 2777–2836, 2009.

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

[16] S. Macskassy and F. Provost, "Classification in networked data: A toolkit and a univariate case study," *Journal of Machine Learning Research*, vol. 8, pp. 935–983, 2007.

[17] G. Mann and A. McCallum, "Simple, robust, scalable semi-supervised learning via expectation regularization," in *Proc. of ICML*, 2007, pp. 593–600.

[18] L. K. McDowell, "Relational Active Learning for Link-Based Classification," in *Proc. of DSAA*, 2015.

[19] J. J. Pfeiffer III, J. Neville, and P. N. Bennett, "Composite likelihood data augmentation for within-network statistical relational learning," in *Proc. of ICDM*, 2014.

[20] J. Neville and D. Jensen, "A bias/variance decomposition for models using collective inference," *Machine Learning Journal*, vol. 73, no. 1, pp. 87–106, 2008.

[21] H. Eldardiry and J. Neville, "Across-model collective ensemble classification," in *Proc. of AAAI*, 2011.

[22] ——, "An analysis of how ensembles of collective classifiers improve predictions in graphs," in *Proc. of CIKM*. ACM, 2012, pp. 225–234.