Relational Active Learning for Link-Based Classification

Luke K. McDowell
U.S. Naval Academy, Annapolis, Maryland U.S.A.
Email: lmcdowel@usna.edu

***Abstract*—Many information tasks involve objects that are explicitly or implicitly connected in a network (or graph), such as webpages connected by hyperlinks or people linked by "friendships" in a social network. Research on *link-based classification* (LBC) has shown how to leverage these connections to improve classification accuracy. Unfortunately, acquiring a sufficient number of labeled examples to enable accurate learning for LBC can often be expensive or impractical. In response, some recent work has proposed the use of active learning, where the LBC method can intelligently select a limited set of additional labels to acquire, so as to reduce the overall cost of learning a model with sufficient accuracy. This work, however, has produced conflicting results and has not considered recent progress for LBC inference and semi-supervised learning. In this paper, we evaluate multiple prior methods for active learning and demonstrate that none consistently improve upon random guessing. We then introduce two new methods that both seek to improve active learning by leveraging the link structure to identify nodes to acquire that are more representative of the underlying data. We show that both approaches have some merit, but that one method, by proactively acquiring nodes so as to produce a more representative distribution of known labels, often leads to significant accuracy increases with minimal computational cost.**

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.

Early work work on LBC typically assumed that a model could be learned on a fully-labeled training graph, while inference was performed on a separate "test graph." More recent work has often considered an alternative scenario, where learning and inference are both performed on a single, partially-labeled graph. In both cases, however, acquiring a significant number of training labels, as needed for learning, can often be prohibitively expensive or impossible. For instance, "labeling" a node may require hours of work by a human annotator (for documents), expensive biological experiments (for protein interactions), or even in-field investigations (for fraud identification). Thus, the cost of acquiring labels can greatly limit the applicability and/or accuracy of LBC.

Observe, however, that while acquiring the labels can be very difficult, often collecting the node attributes and basic link structure of the graph is easy (e.g., for social and webpage networks) [3], [4]. This has inspired two complementary lines of research that both seek to reduce the number of training labels that are needed. First, some studies have applied some form of semi-supervised learning (SSL), for instance using the provided labels to help predict labels for all unlabeled nodes, and then learning a new model based on both known and predicted labels [5], [6], [7]. Second, a few studies have applied "active learning" methods to LBC [8], [9], [10], [11]. In this approach, the LBC method has a budget B that can be used to "purchase" the true labels of some nodes (for instance by paying a human annotator), and the goal is to identify the most profitable nodes to purchase labels for.

Unfortunately, the results of these studies with active learning present a number of problems that restrict their use. First, none of the prior studies have examined the most effective forms of semi-supervised LBC. For instance, some approaches either used no SSL [11] or a relatively weak single round of SSL [8]. Other work [9] used effective SSL but did not use any strategies to prevent relational bias from causing "flooding" of one particular label across large parts of the network during inference [12], [13], which recent work has demonstrated can be a significant problem with semi-supervised LBC [6], [7]. Second, none of these prior studies considered the use of the *attributes* of linked nodes (i.e., "neighbor attributes") as model features, as opposed to using the *labels* of linked nodes ("neighbor labels"). While relying on neighbor labels has been typical for almost all work with LBC, recent work [14] has shown that using neighbor attributes can substantially increase accuracy when training labels are sparse. However, because such models lead to very different forms of inference, prior findings for active learning with neighbor labels may not carry over to models that use neighbor attributes instead.

Finally, the prior studies of active learning for LBC have produced very conflicting results. For instance, Bilgic et al. [8] found that acquiring the labels of nodes with high prediction uncertainty lead to increased accuracy. In contrast, Kuwadekar and Neville [9] found, surprisingly, that acquiring the labels

of nodes with the *least* prediction uncertainty performed best. Thus, active learning has not been evaluated with the most accurate types of LBC, and even for the comparisons that have been made there remains substantial confusion regarding the most effective types of active learning to employ.

Our contributions are as follows. First, we present the first study of active learning (AL) for LBC that considers recent improvements to LBC with SSL, neighbor attributes as features, and corrections for relational bias. We show that, when these improvements are included, none of the existing AL methods fare well vs. random guessing. Second, we examine why AL is so complex for relational domains like LBC. Specifically, we argue that an ideal AL method would select nodes that were *informative* compared to the existing information, *influential* on learning and inference, *distributed* throughout the network, and *representative* of the underlying data. We explain multiple existing AL methods in terms of the degree to which they meet these competing objectives. Third, inspired by our observations of problems with existing methods, we introduce two novel AL methods, CLASSSEEK and UNC+LINKEVID. Both methods leverage the link structure to effectively identify a more *representative* set of nodes to acquire. Fourth, we evaluate the new methods on three real datasets. We show that both lead to some improvements in accuracy, but that CLASSSEEK in particular, by proactively acquiring nodes so as to produce a better distribution of known labels, often leads to significant accuracy gains compared to competing approaches. Finally, we use two ablation studies to better understand why CLASSSEEK improves accuracy. We show that its distribution-balancing strategy and its use of degree-biased node sorting both contribute to its gains, but that refinements to the sorting could be fruitful future work.

II. BACKGROUND AND RELATED WORK

This section first describes the general problem of LBC and how it can be performed when given a set of nodes with known labels. Next, we briefly summarize existing work with active learning for both traditional (non-LBC) and LBC settings.

A. 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 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`."

LBC models can be characterized based on the kinds of features that they use. For instance, an "attribute only" model uses only self attributes; this is a common baseline. 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) [15]. CI methods include Gibbs sampling, belief propagation, and ICA (Iterative Classification Algorithm) [12].

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 ICA, a simple, popular, and effective algorithm [12], [8], [6]. ICA first predicts 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.

RI has been used for LBC much less frequently, due in part to early work that showed that RI sometimes hurt accuracy compared to using attributes alone [1] and/or generally underperformed CI [15]. However, a more recent study showed that RI can lead to significantly higher accuracy than CI — *if* only a small number of labels are available for training [14]. For this reason, we consider both RI and CI in this paper. Notably, they may have very different behaviors, since CI requires collective inference and RI does not.

B. Active Learning

Active learning (AL) [16] is an approach to learning where the set of training data is not fixed, but where the AL method can select some examples to acquire or "purchase" the labels for, e.g., by submitting to an "expert." Like related work for LBC, we focus on the "pool-based" scenario, where there is an initial set of labeled examples L and then a set of unlabeled examples (the pool P) that the AL method may select from. During each round, the AL method may select k examples from P and add them to L. The new L is then used for learning and inference, helping to inform the selections that are made during the next round. Typically, this process involves computing a utility score for each example in P, then sampling or selecting k examples based on their utilities.

For instance, many AL methods involve variants of uncertainty sampling [17], based on the intuition that acquiring the labels of examples for which the learner currently has high

prediction uncertainty is likely to substantially affect subsequent learning. Uncertainty sampling has been frequently used, though it can sometimes be unduly influenced by noise and outliers [18]. Some work has considered improving uncertainty sampling by weighting examples based on how representative they are of the data, or by seeking to exclude outliers [19].

C. Active Learning with LBC

There are many domains where the basic attributes and link structure of the nodes are easy to obtain (e.g., for web pages, documents, social networks, etc.), but where acquiring the associated labels of interest for these nodes is much more expensive. As a result, some prior studies have studied active learning for relational data [8], [20], [9], [11], [10], [21]. We discuss the most relevant such work below.

ALFNET [8] first uses modularity clustering to cluster the nodes based only on their link structure. During each round, ALFNET runs LBC, then scores each node based on the results. Each node v's score reflects the extent to which there is *disagreement* between (a) the prediction of the relational (i.e., link-aware) classifier for v, (b) the prediction of a (link-unaware) model that uses only a node's attributes for v, and (c) the most commonly predicted label within v's cluster. Clusters are, in turn, scored based on the average disagreement of all nodes in that cluster, normalized by the number of currently "known" nodes in that cluster. Next, the k clusters with highest scores are selected. Finally, within each such cluster, the node with the highest disagreement score is chosen for labeling.

Bilgic et al. [8] reported higher accuracies from learning with ALFNET compared to uncertainty or random sampling, using two real datasets (Cora and Citeseer). We evaluate their approach in Section VII, and also include those same datasets.

ALFNET sought to identify nodes with high prediction uncertainty that were surrounded by other nodes with high uncertainty. Thus, it was, in part, a natural extension of uncertainty sampling to leverage the link structure. In contrast, later work by Kuwadekar and Neville [9] found that selecting nodes that had *low* prediction uncertainty produced better learning than selecting nodes with high uncertainty. Specifically, they computed, for each node, a *weighted density disagreement* (WDD) score which (somewhat contrary to its name) produces higher scores for a node v when (a) v's label predictions *diverge* from the mean overall prediction (implying greater confidence in the prediction) and/or (b) v's label predictions *agree* with the prediction of its linked neighbors. They noted that this was contrary to the typical results with uncertainty sampling, and conjectured that this effect might be due to the helpful effects of label propagation during collective inference. We evaluate the WDD metric in Section VII.

Other studies have focused more exclusively on selecting nodes based on the link structure [20], [10], [11], ignoring any attributes. For instance, Kajdanowicz et al. [11] considered metrics such as betweeness and page rank, but found that such metrics often performed little better than random guessing, especially when there was a significant number of class labels.

Using active learning for LBC generally implies the existence of a set of unlabeled nodes that interlink with the labeled nodes. In this context, the use of effective semi-supervised learning can be essential to improving learning, because of the difficulty that LBC models, particularly CI, have with learning the parameters for label-based features when the data is only sparsely-labeled. Thus, we use recent improvements to learning and inference for LBC, including EM-like learning from predicted labels [5], [6], [22], along with corrections for relational bias in order to prevent "flooding" during inference [6], [7] (see Section VI for details). This ensures that we are improving upon state-of-the-art results, rather than upon the accuracy of less accurate methods.

III. Designing Active Learning for LBC

Above, we discussed some AL strategies, particularly where applied to LBC. Most of the aforementioned methods focus on one or two particular objectives, such as identifying nodes for which the classifier lacks sufficient information to make a confident prediction. Ideally, however, AL should consider a larger range of objectives. In particular, we propose that the ideal active learner would select nodes for labeling that were simultaneously **Informative, Influential, Distributed,** and **Representative**. Table I evaluates the extent to which five existing methods and two new methods (discussed later) achieve these objectives. For each method and objective, we give a subjective rating of High (the method is likely to fare well for this objective), Low (the converse), or Mid (fairly neutral in regards to this objective). Below, we explain these objectives in relation to the AL methods of Table I.

- **Informative.** Uncertainty sampling (UNCRTN) and ALFNET specifically seek to identify nodes for which the classifier cannot make a confident prediction. Such examples should lie close to the classifier's current decision boundary, and thus acquiring labels for such nodes may be *informative* for refining that boundary. In contrast, acquiring labels for nodes with highly confident predictions (as with WDD) is unlikely to be very informative.
- **Influential.** The DEGREE method selects nodes for labeling in proportion to their degree. Because high degree nodes link to many other nodes, they have high *influence* on both learning and inference. In addition, the WDD metric as defined by Kuwadekar and Neville sums the agreement scores of a node's neighbors, thus implicitly favoring the selection of high degree nodes (though they did not discuss this effect). In contrast, methods like UNCRTN that favor nodes with high uncertainty may indirectly favor *low* degree nodes, because with LBC a high degree node often has *low* uncertainty (due to the influence of many neighbors on its predictions). Note that ALFNET also seeks nodes with high uncertainty, but does not have the same implicit degree bias as UNCRTN because it's per-node disagreement formula produces only a few distinct values within each cluster. Thus, in practice random choices are made among the

TABLE I
SUMMARY OF EXISTING (AND PROPOSED) ACTIVE LEARNING METHODS FOR LBC.

Method	The extent to which the AL method select nodes that are... **Informative?**	**Influential?**	**Distributed?**	**Representative?**
RANDOM	**Mid** (rate of correcting prediction errors depends upon overall error rate)	**Mid** (% of high degree nodes chosen depends on their prevalence)	**Mid** (on average, nodes will be well distributed)	**Mid** (good in the limit; though possibly not for small B)
UNCRTN [17], [16]	**High** (seeks uncertain nodes)	**Low-Mid** (low certainty favors low degree nodes?)	**Mid**	**Low-Mid** (favoring uncertainty may cause bias)
DEGREE [23], [11]	**Low-Mid** (high degree nodes less likely to be prediction errors?)	**High** (prefers high degree nodes)	**Mid**	**Low** (favoring high degree may cause bias)
ALFNET [8]	**High** (seeks uncertain nodes)	**Mid**	**High** (picks from distinct clusters)	**Low-Mid** (favoring uncertainty may cause bias)
WDD [9]	**Low** (seeks *certain* nodes)	**High** (metric implicitly favors high degree)	**Mid**	**Low-Mid** (favoring certainty may cause bias)
CLASSSEEK	**Low-Mid** (high degree nodes less likely to be prediction errors?)	**High** (prefers high degree nodes)	**Mid**	**High** (seeks good class distribution) and **Low(?)** (favoring high degree may cause bias)
UNC+LINKEVID	**High** (seeks the "most-surely" uncertain nodes)	**Low-Mid** (low certainty favors low degree nodes?)	**Mid**	**High** (due to evidence for some class) and **Low(?)** (from favoring uncertainty)

nodes within a cluster whose predictions do not agree with their neighbors.

- **Distributed.** Ideally, the AL method would select nodes that are widely *distributed* throughout the network. This ensures their influence during learning and is especially useful for CI for "seeding" the collective inference process with stable, true labels in distinct parts of the network. Most AL methods are neutral in regards to this objective, except that ALFNET boosts distributedness by explicitly selecting from different clusters.
- **Representative.** For learning, the training set should ideally be mostly comprised of nodes that are fairly *representative* of the overall population. Prior work [18] has noted that uncertainty-based methods (such as UNCRTN and ALFNET) may fare poorly in this regard, since nodes with high uncertainty may be outliers rather than "typical" examples. Likewise, the nodes selected by WDD may not be highly representative, since nodes with very *low* uncertainty are also not typical.

Table I shows that each method has relative strengths and weaknesses, and there are potential avenues for (and existing work) on improving many of them. We note, however, that none of the existing AL methods for LBC discussed in this paper fare very well with selecting *representative* nodes. While ensuring representative examples is a challenge also for non-LBC active learning domains, with LBC the link structure of the network presents a particular challenge and opportunity.

The next two sections propose two new techniques for answering this challenge. The first technique uses the links, indirectly, to help select nodes that well-represent the class distribution of the underlying population. The second technique transforms uncertainty measures, via the links, to select nodes that are both uncertain *and* that are hopefully more representative because they exhibit strong characteristics suggestive of a particular class.

IV. ACTIVE LEARNING WITH CLASSSEEK

An important aspect of *representative* node selection, though one not previously considered with LBC, is the class distribution of the labeled nodes V^K. Specifically, V^K should contain a good diversity of examples within each class, *and* a mixture of classes that reflects the underlying distribution. However, in preliminary experiments, we observed that methods that favored low certainty, high certainty, or high degree often preferentially selected the nodes of some classes more frequently than others, resulting in a biased set V^K.

Assume that we have an estimate D^* of the actual class distribution of the example population. Such an estimate could be derived based on background demographic information (when predicting age or gender), results from previous prediction tasks (e.g., when predicting fraud or system failures), the results of online and/or anonymous surveys, or from an initial series of randomly acquired labels. Could we use such an estimate to guide an AL method so that it selects nodes that closely reflect the underlying class distribution, *yet* without substantially reducing the extent to which the chosen nodes V^K are informative, influential, and distributed?

Figure 1 presents the pseudocode for a new algorithm, CLASSSEEK, that seeks to select a class-representative set of nodes for labeling, while being combined with another method such as RANDOM, DEGREE, or UNCRTN. The key idea is to ask, at each step: which class c, if added to the distribution of the current known labels Y^K, would yield a distribution most similar to the desired one? Next, select some node v that (hopefully) has that true label.

More specifically, CLASSSEEK operates as follows. Steps 2-3 learn a relational model based on the current known nodes'

ClassSeek_learn $(V, E, X, Y^K, B, D^*, \mathcal{U})$=
// V=nodes, E=edges, X=attribute vectors, Y^K=initial known labels,
// B=budget, D^*=population class distr. estimate, $\mathcal{U}$=utility function

1 **repeat**
2 $M \leftarrow SSL\_learn(V, E, X, Y^K)$
3 $Y^U \leftarrow predictLabels(V, E, X, Y^K, M)$
4 $D \leftarrow computeLabelDistribution(Y^K)$
5 $c \leftarrow selectBestClass(D, D^*)$
6 $\mathcal{O} \leftarrow orderNodesByUtility(V, E, Y^K, Y^U, \mathcal{U})$
7 **repeat**
8 $v \leftarrow dequeue(\mathcal{O})$
9 **until** $Y^U_v = c$ // i.e., found node with desired predicted label
10 $V^U = V^U \setminus \{v\}$
11 $V^K = V^K \cup \{v\}$
12 $Y^K = Y^K \cup \{acquireTrueLabel(v)\}$
13 **until** $|\mathbf{Y^K}| = \mathbf{B}$ // i.e., budget exhausted

Fig. 1. Generic CLASSSEEK algorithm for active learning.

labels Y^K, then predict new labels (using CI or RI) for all unknown nodes V^U. Steps 4-5 identify the "best" class to, ideally, add to Y^K; we select the class c that minimizes the Euclidean distance between D and D^*. Step 6 orders the nodes in V^U according to some metric (discussed further below), then steps 7-9 select the first node v in the resultant list $\mathcal{O}$ that has a *predicted* label of c. Finally, steps 10-12 acquire the true label for v and update V^U, V^K, and Y^K as appropriate.

The CLASSSEEK algorithm is simple to describe, but how well it works in practice depends on a few significant issues. First, given that it does not know the true labels of the nodes in V^U, will this method actually result in a distribution D that is similar to D^*? Second, how should the utility/ranking in step 6 be computed, and what impact does that have on how informative, influential, distributed, and representative the final V^K is? These questions are inter-related.

For this paper, we chose to sort the nodes in step 6 based solely on their degree, but via probabilistic sampling. Specifically, the output list $\mathcal{O}$ is initially empty and each node has chance proportional to its degree of being added next to $\mathcal{O}$. Thus, a high degree node is more likely to appear early in $\mathcal{O}$, and thus more likely to be ultimately selected by step 9 (if it's predicted label matches c). Thus, as shown in Table I, CLASSSEEK is rated "High" in influentialness (due to preferring high degree nodes). CLASSSEEK also fares well for being representative, because we find that the node selection of steps 6-9 *does*, often, select nodes with the desired class label. This is due to several factors. First, the label prediction Y^U_v that is used to filter out undesirable nodes is based on the result of inference with RI or CI, and thus reflects the influence of linked neighbors. Using such neighbors improves accuracy, and often especially so for higher degree nodes. Second, sorting with a preference for higher degree nodes does *not* negatively impact the accuracy of the predicted labels for nodes near the front of the list $\mathcal{O}$ (in contrast to the results obtained if the sorting prefers more uncertain nodes, whose predictions are likely *not* to be correct). Finally, CLASSSEEK need not succeed "every time" in correctly selecting a node with class c. If the true label of v turns out to be a different class, v is still added to V^K (providing useful information). Moreover, during the next iteration the algorithm will select the same c and try again. As long as the true label "often" matches the desired label, the final distribution of Y^K will tend towards the desired D^*.

V. ACTIVE LEARNING WITH UNC+LINKEVID

This section considers a recent development in non-LBC active learning, then explain how it can adapted for LBC. In this section, let m be the most likely predicted class for some example x (or node v_i), and n the next most likely class. Also, let $\mathcal{N}_i$ be the set of nodes that are adjacent to node v_i, i.e., in the "neighborhood" of v_i Furthermore, let $X_{\mathcal{N}_i}$ be the set of attribute vectors for all nodes in $\mathcal{N}_i$ ($X_{\mathcal{N}_i} = \{\vec{x_j} | v_j \in \mathcal{N}_i\}$).

Recently, Sharma and Bilgic [24], working in the context of i.i.d. (non-LBC) data, described a modification to uncertainty sampling. They noted that an example x may have high prediction uncertainty for two reasons: either x's attributes provide only weak evidence for any class (the "least-surely uncertain") or its attributes provide strong evidence for more than one class (the "most-surely uncertain"). They show how to distinguish such cases via an "evidence function" $E(x)$:

$$E(x) = E^m(x) \times (-E^n(x))$$

where $E^m(x)$ and $E^n(x)$ are the "evidence" (from the attributes) for x belonging to class m or n, respectively. Most-surely nodes have high uncertainty and high $E(x)$, while least-surely nodes have high uncertainty and low $E(x)$.

They then propose the following AL method: select the top N examples with maximal uncertainty, then from these N select for labeling the one with maximal $E(x)$ (the most surely uncertain). They show using several datasets that this approach significantly improves accuracy compared to uncertainty sampling or to selecting the "least-surely uncertain" examples.

This "most surely" method does not directly apply to LBC, since it assumes that predictions are based solely on a fixed-length attribute vector, whereas LBC must also incorporate the influence of a (varying) number of neighbors. However, the method is intriguing, since its gains may result from identifying examples that are more likely to be strongly representative of a particular class, which might benefit LBC.

In response, we adapted the "most-surely" method to our LBC context. The key change necessary is converting $E^m(x)$ and $E^n(x)$ to appropriate $E^m(v_i)$ and $E^n(v_i)$ (for some node v_i). This requires examining the details of how predictions are made for node v_i, including the impact of its linked neighbors. Adapting Equation 3 from [14] we have[1]

$$p(y_i = c|\vec{x_i}, X_{\mathcal{N}_i}) \propto p(y_i = c|\vec{x_i}) \prod_{v_j \in \mathcal{N}_i} \frac{p(y_i = c|\vec{x_j})}{p(y_i = c)}. \quad (1)$$

Observe that each neighbor v_j provides a product term that provides stronger evidence either for m *or* for n, depending on the specific probabilities. Let $\mathcal{ML}_i$ be the set of neighbors

[1]This derivation is for RI; for CI, replace $p(y_i = c|\vec{x_j})$ with $p(y_i = c|y_j)$.

of v_i that provides evidence supporting m over n (i.e., the *M*-evidence based on *Links*). This can be computed as:

$$\mathcal{ML}_i \triangleq \{v_j | v_j \in \mathcal{N}_i \wedge \frac{p(y_i = m|\vec{x_j})/p(y_i = m)}{p(y_i = n|\vec{x_j})/p(y_i = n)} > 1\}$$
$$= \{v_j | v_j \in \mathcal{N}_i \wedge ln\frac{p(y_i = m|\vec{x_j})/p(y_i = m)}{p(y_i = n|\vec{x_j})/p(y_i = n)} > 0\}.$$

The set $\mathcal{NL}_i$ (of neighbors supporting n instead) is defined identically, except the inequality is replaced with < 0. Together, these will handle the product terms from Equation 1. We now consider the remaining term, $p(y_i = c|\vec{x_i})$. This is a simple vector-based classifier; Bilgic and Sharma discuss how to handle several possible models. We will use logistic regression (LR), and adapt their two-class solution to support multiple classes (see [24] for the initial derivations). Let $\mathcal{MA}_i$ be the set of attributes' numbers that give evidence for m vs. n, $x_{i,l}$ the value of the l'th attribute for v_i, and θ_m the weight vector learned by LR for class m. Then

$$\mathcal{MA}_i \triangleq \{l | x_{i,l}(\theta_{m,l} - \theta_{n,l}) > 0\}.$$

Replace the > 0 test with < 0 to obtain $\mathcal{NA}_i$ instead of $\mathcal{MA}_i$. Given these definitions, we can compute $E^m(v_i)$, the overall evidence for node v_i having class m, considering the influence of its own attributes and its neighbors, as follows:

$$E^m(v_i) = \sum_{l \in \mathcal{MA}_i} x_{i,l}(\theta_{m,l} - \theta_{n,l}) + \sum_{v_j \in \mathcal{ML}_i} ln\frac{p(y_i = m|\vec{x_j})/p(y_i = m)}{p(y_i = n|\vec{x_j})/p(y_i = n)}.$$

$E^n(v_i)$ is computed with the same formula, but replacing $\mathcal{MA}_i$ and $\mathcal{ML}_i$ with $\mathcal{NA}_i$ and $\mathcal{NL}_i$. Finally,

$$E(v_i) = E^m(v_i) \times (-E^n(v_i)).$$

We use this method in the same style suggested by Sharma and Bilgic, adapted for a batch size of 5. Specifically, for each round we select the 40 nodes with the highest uncertainty, then from these select the 5 "most-surely uncertain" (highest $E(v_i)$). We call this method UNC+LINKEVID, because it combines uncertainty sampling with a link-aware evidence score. Ideally, this may improve the representativeness of the acquired nodes; Section VII evaluates its impact.

VI. EXPERIMENTAL METHOD

A. Datasets and Features

We used three commonly-used real datasets (see Table II). **Cora** (cf., [12]) is a collection of machine learning papers and **Citeseer** (cf., [12]) is a collection of research papers. Attributes represent the presence of certain words, and links indicate citations. We mimic Bilgic et al. [8] by ignoring link direction, and also by using the 100 top attribute features after applying PCA to all nodes' attributes. **Gene** (cf., [15]) describes the yeast genome at the protein level. Links represent protein interactions. We mimic Xiang & Neville [5], [6] and predict protein localization using four attributes: Phenotype, Class, Essential, and Chromosome.

TABLE II
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%

B. Classifiers

Both CI and RI require learning a classifier to predict the label based on self attributes, and (for RI) a separate classifier based on neighbor attributes. We use logistic regression (LR), because it usually outperformed other alternatives [12], [8], [14]. To combine the predictions from a varying of number of neighbors, RI used the technique shown in Equation 1.

CI also requires a classifier to predict a node's label based on neighbor labels. McDowell & Aha [6] 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 self attributes (described above), using the "hybrid model" method [6], similar to Equation 1.

C. Learning and Collective Inference

For inference and semi-supervised learning, we chose methods that performed well in prior comparisons [12], [8], [6]. CI used 10 iterations of ICA for collective inference. For learning, we use 10 iterations of SSL-EM [5], [6], [7]: repeatedly predict the labels (with CI or RI) for all unknown nodes, then relearn models using predicted and known labels.

Prior work with LBC has often reported problems with "flooding," where large regions of the network end up having the same predicted label. Recent work has studied how to correct the relational bias that leads to such flooding, e.g., by modifying learning with label regularization [6] or inference with a "maximum entropy" constraint [7]. In our work, we use a similar inference correction, *class mass normalization* (CMN) [25]. CMN rescales the predicted probabilities of every node so that the total mass of each class matches the desired label distribution, which can be estimated from the known labels, thus making "over-represented" classes in the predicted labels less likely to be selected. We found this yielded gains very similar to those provided by label regularization, and that using one of those two methods was important for obtaining the best accuracy, especially with CI.

D. Evaluation Procedure

We measure performance via accuracy, averaged over 40 trials for each combination of an AL method and dataset. For each trial, we first randomly selected 10% of the nodes to be the "evaluation set" V^E, and also ten nodes to be the initial set of known nodes, V^K. The remaining nodes, $V^U = V \setminus V^E \setminus V^K$, are the "unknown nodes." Only nodes in V^U (*not* those in V^E) could be selected for acquisition by the AL method, and accuracy was evaluated *only* on V^E.

TABLE III
STATISTICAL ANALYSIS FOR FIGURE 2, COUNTING SIGNIFICANT WINS, LOSSES, AND TIES FOR CLASSSEEK COMPARED TO 4 OTHER METHODS (DETAILS IN SECTION VI-D). THE COUNTS ARE COMBINED FOR RI AND CI, FOR $10 < B < 100$.

CLASSSEEK VS.	Cora			Citeseer			Gene		
	W	L	T	W	L	T	W	L	T
... ALFNET	31	0	3	29	0	5	10	2	22
... UNCRTN	34	0	0	33	0	1	6	1	27
... WDD	30	0	2	33	0	1	22	1	11
... RANDOM	33	0	1	34	0	0	12	0	22

The AL proceeds in rounds. After being provided with the initial 10 node labels, during each round the AL method selects five additional nodes for an "expert" to label. During the next round, those five nodes are then added to the set V^K and their labels are available for learning and inference.

We focus on the important sparsely-labeled case [4], [3], [6], e.g., when less than 10% of the nodes will be labeled. Specifically, we evaluate cases where the total budget $B \in \{15, 20, ..., 95\}$. If $B = 25$, then 10 nodes' labels are from the initial random selection and the next 15 are chosen by AL. To assess results, we mimic Bilgic et al. [8] and use paired t-tests with a 10% significance level. Like them, we aggregate over the values of B. Specifically, when comparing method E vs. F, we count the number of times that E has accuracy that is *significantly* higher than F (a win), lower than F (a loss), or neither (a tie). We aggregate over the 17 values of B in $\{15, 20, ..., 95\}$. Where RI and CI have similar trends, we also combine their respective counts, and thus the best performance possible for method E would be 34 wins vs. method F.

E. Methods Considered

We consider the five existing methods listed in Table I and the two new methods (CLASSSEEK and UNC+LINKEVID). Let δ be the "margin of confidence" for a node, i.e., the difference in the classifier's output for the most likely label vs. for the next most likely label. UNCRTN favors nodes with *low* values of δ; we select a node with probability proportional to $1-\delta$, as opposed to deterministically selecting the nodes with minimal δ, which is known to exhibit problems [16]. Likewise, DEGREE selects a node with probability proportional to its degree. By default, CLASSSEEK uses the same degree-based technique to select the nodes for its ordered list $\mathcal{O}$.

We evaluate ALFNET [8] using code obtained from the authors. We used WDD as described by Kuwadekar and Neville [9]. They also used an ensemble of classifiers with WDD. We did not do so, in order to more closely compare each method based on the basic metrics used; future work should evaluate the impact of this choice.

VII. RESULTS

We first compare CLASSSEEK and UNC+LINKEVID against the existing AL methods. We find that CLASSSEEK often significantly outperforms the other methods. Thus, we then present two ablation studies that examine this method.

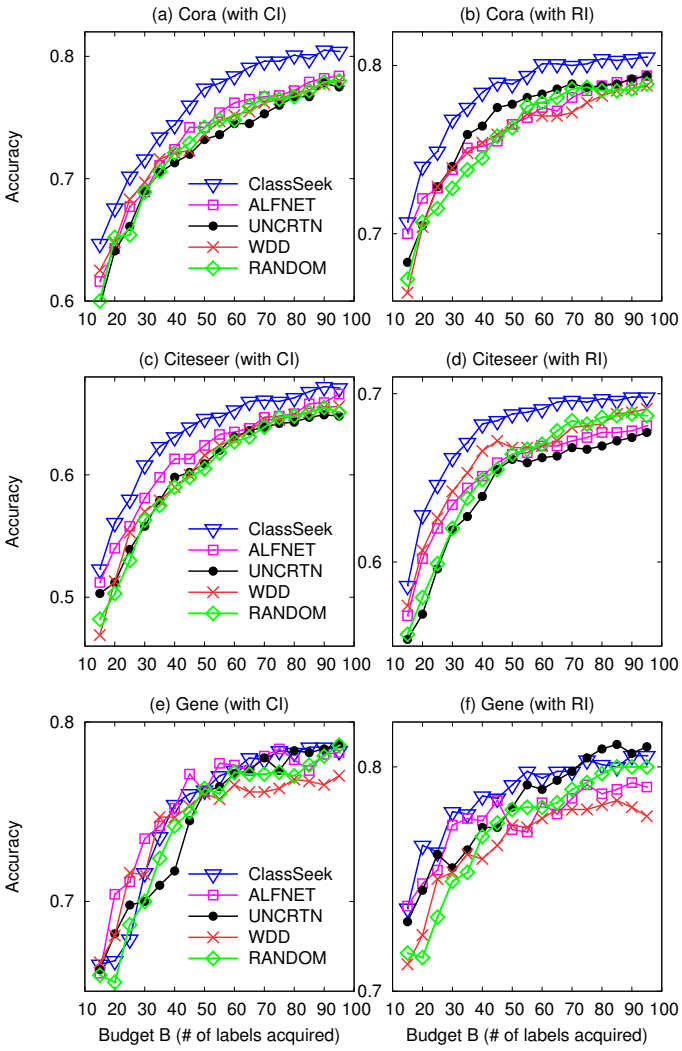

Fig. 2. Average accuracy for one of the new AL methods (CLASSSEEK) vs. four existing methods. Table III provides statistical comparisons for some of these methods. Figure 3 compares some of these methods vs. UNC+LINKEVID. Note that the y-axes vary, since our focus is not on comparing CI vs. RI, but on examining the relative behavior of different AL methods.

Evaluating the New Active Learning Methods: Figure 2 compares CLASSSEEK against four of the existing methods from Table I (DEGREE is discussed later). CLASSSEEK clearly performs best for Cora and Citeseer, with either CI or RI. It is also generally the best or one of the best for Gene with RI, while no method performs consistently well for Gene with CI. Table III provides statistical analysis for these results, showing that CLASSSEEK has many significant "wins" vs. the other methods in almost all cases for Cora and Citeseer, and in about a third of the cases for Gene (with very few "losses").

For figure clarity, we show results with UNC+LINKEVID separately, in Figure 3, and to conserve space we show only some the representative results. Compared to UNCRTN and RANDOM, UNC+LINKEVID performs very poorly with RI on Gene (see Figure 3(d)). However, in almost all other cases,

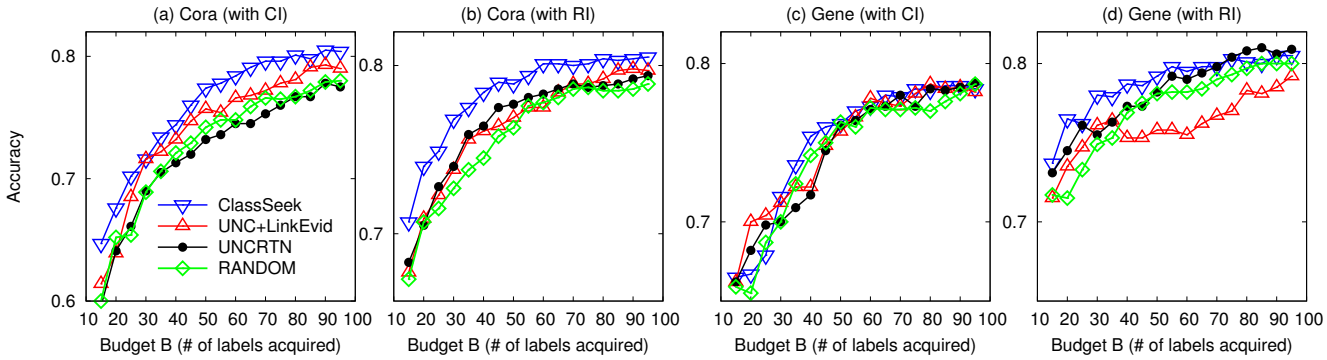

Fig. 3. Accuracy comparing the two new methods (CLASSSEEK and UNC+LINKEVID) against two existing methods. To conserve space, results are shown for only some combinations of the datasets with CI/RI. Note that with Gene the choice of CI vs. RI greatly affects the relative performance of the AL methods.

TABLE IV
ANALYSIS COMPARING UNC+LINKEVID VS. SEVERAL ALTERNATIVES, SHOWING COUNTS WITH CI (AT TOP) AND RI (AT BOTTOM). FIGURES CORRESPONDING TO SOME OF THE MODEL+DATASET PAIRS GIVEN BELOW ARE SHOWN IN FIGURE 3.

UNC+LINKEVID vs. ...	Cora W	Cora L	Cora T	Citeseer W	Citeseer L	Citeseer T	Gene W	Gene L	Gene T
(using CI)									
... CLASSSEEK	0	11	6	0	14	3	1	1	15
... UNCRTN	12	0	5	1	0	16	0	0	17
... RANDOM	6	0	11	1	0	16	2	0	15
(using RI)									
... CLASSSEEK	0	15	2	0	17	0	0	14	3
... UNCRTN	1	0	16	16	0	1	0	12	5
... RANDOM	4	0	13	0	0	17	1	10	6

UNC+LINKEVID succeeds in improving accuracy (averaged over all values of B) compared to UNCRTN and RANDOM. Thus, our attempt to improve upon uncertainty sampling, via the "most-surely certain" idea, was at least partially successful; its best gains are shown in Figure 3(a). However, most of these gains are fairly small, and thus the statistical comparison in Table IV shows very few wins for UNC+LINKEVID. Moreover, CLASSSEEK very frequently beats UNC+LINKEVID (for 72 of the 102 possibilities shown in Table IV). Thus, we focus on CLASSSEEK in the remainder of this paper.

Varying the node ordering used by CLASSSEEK: We next compare CLASSSEEK against several variants that use the same basic algorithm shown in Figure 1, but use a different sorting/utility function in step 6, instead of sampling based on degree. $+Rand$ (short for CLASSSEEK$+Rand$) randomly sorts the nodes, while $+Margin$ select nodes based on $1-\delta$ uncertainty (see Section VI-E). $+Cluster$ uses the same modularity clustering employed by ALFNET, then scores each cluster based on their size divided by the number of known nodes they contain. Nodes are sorted based on the score of their cluster (and then randomly within the same cluster). Only one node from each cluster may be selected within each round.

Using $+Rand$ removes any degree-based bias, $+Margin$ seeks to improve the informativeness of the selected nodes, and $+Cluster$ seeks to improve the distributedness of the nodes. Figure 4 and Table V show the results. Overall, the original CLASSSEEK (with degree-based sampling) appears most successful. For instance, it has numerous wins compared to the others for Cora and Citeseer, and very few losses in any case. Gene is somewhat of an exception. Here, $+Cluster$ sometimes has the highest accuracy with CI, and $+Rand$ often has the highest accuracy with RI. Table V shows these differences sometimes lead to significant "losses" for CLASSSEEK. Nonetheless, neither of these methods performs consistently well with the other dataset/model combinations; instead CLASSSEEK is the most consistent overall performer.

Varying the class balancing method of CLASSSEEK: We next compare CLASSSEEK against two variants that both begin by ordering the nodes by probabilistically sampling based on node degree (as with CLASSSEEK), but then use different conditions in step 7-9 of the algorithm to determine when to actually select a proposed node (from the queue) for labeling. The CLASSSEEK-KNOWN variant makes the following observation: frequently, nodes are very likely to link to other nodes with the same label [26]. Thus, instead of selecting nodes that are *predicted* (with the classifier) to have node c, it selects the first node encountered that *links* to at least one node with *known* label c. Alternatively, DEGREE is a natural comparison that starts with the same node ordering as CLASSSEEK, but then makes no attempt to balance the class distribution (i.e., it simply selects the first nodes in the list $\mathcal{O}$).

Figure 5 and Table VI show the results. CLASSSEEK dominates, with many significant "wins." These results illustrate (a) the additional value of CLASSSEEK's class balancing method, compared to using node degree alone and (b) the superiority of using label predictions from the CI or RI classifier, compared to directly using the links. The latter effect can be explained in part by noting that CLASSSEEK-KNOWN introduces even more bias in favor of very high degree nodes (since they are more likely to link to *some* known node with the desired label), and thus decreases representativeness.

Discussion: Figure 2 shows that none of the existing methods consistently beat RANDOM. In particular, ALFNET, UN-

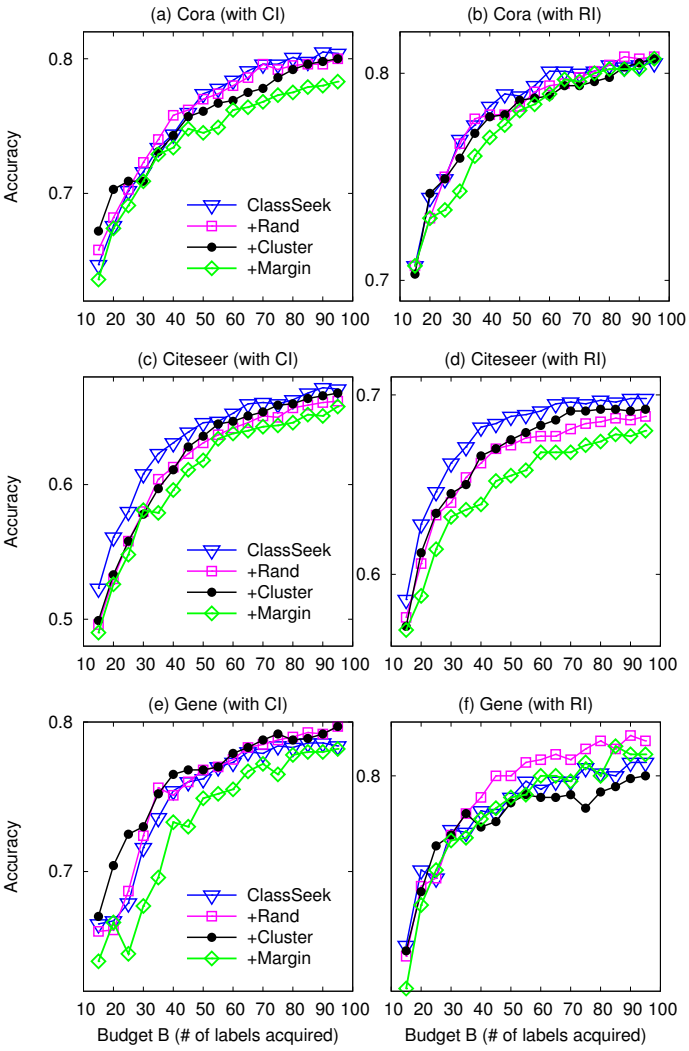

Fig. 4. Ablation study: comparision of CLASSSEEK (with degree-based ordering) vs. variants that use a different node ordering/sorting. Table V provides relevant statistical analysis.

TABLE V
STATISTICAL ANALYSIS OF RESULTS OF FIGURE 4, COMPARING CLASSSEEK VS. ALTERNATIVES. COUNTS ARE COMBINED FOR RESULTS WITH CI AND RI.

CLASSSEEK vs.	Cora			Citeseer			Gene		
	W	L	T	W	L	T	W	L	T
... +Rand	0	0	34	29	0	5	0	7	27
... +Cluster	7	2	25	18	0	16	1	3	30
... +Margin	12	0	22	18	0	16	2	3	29

CRTN, and WDD all had significant wins vs. RANDOM in only 7 or 8 out of 102 possible cases (results not shown), whereas CLASSSEEK had 79 such wins (see Table III). In contrast, Bilgic et al. [8] previously found that ALFNET outperformed RANDOM in at least half of the cases considered, with Cora and Citeseer. Likewise, Kuwadekar and Neville [9] reported that their active learning with WDD consistently outperformed RANDOM, UNCRTN, and ALFNET.

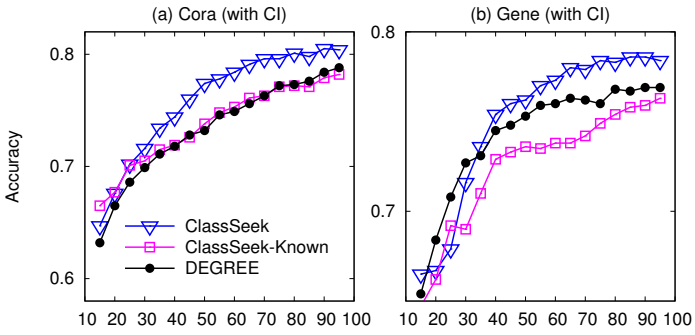

Fig. 5. Comparision of CLASSSEEK vs. variants that use degree-biased selection but a different (or no) label-balancing method. Results with Citeseer were similar to those with Cora, and results with RI showed similar trends (not shown due to lack of space).

TABLE VI
ANALYSIS OF RESULTS OF FIGURE 5, COMPARING CLASSSEEK VS. ALTERNATIVES, COMBINED FOR CI AND RI.

CLASSSEEK vs. ...	Cora			Citeseer			Gene		
	W	L	T	W	L	T	W	L	T
DEGREE	31	0	3	34	0	0	24	0	10
CLASSSEEK-KNOWN	28	0	6	32	0	2	28	0	6

Multiple explanations are possible for these conflicting results. As noted in Section I, Bilgic et al. used a relatively weak "single-pass" form of learning, as opposed to the EM-like approach that we used (see Section VI-C), which prior work has shown to produce better accuracy [6]. Likewise, they did not use a technique like CMN to prevent flooding (see Section VI-C), which we found substantially improved accuracy. Thus, it may be that the performance differences they observed are greatly diminished when stronger learning and inference choices are made; this emphasizes the importance of our using the most recent approaches for correcting relational bias and for applying semi-supervised learning with our results. Kuwadekar and Neville also did not use a procedure like CMN. Moreover, they evaluated performance by measuring the results of simulated "ceiling" inference, which eliminates the effect of collective inference. Such an approach is useful for measuring AL's impact on *only* learning, but clearly affects the results by eliminating the possibility of flooding during inference, and eliminating the utility of distributing labeled nodes throughout the network to aid inference (as provided by ALFNET's clustering). Their use of ensemble learning (see Section VI-E) may also have influenced the relative results.

Ideally, an AL method would select nodes that were simultaneous informative, influential, distributed, and representative. In practice, these objectives are sometimes conflicting, and understanding the relative tradeoffs can help to understand the behavior of different AL methods. Consider, for instance, the results for the $+Margin$ variant shown in Figure 4. Using $+Margin$ *does* lead to greater informativeness, as would be expected to arise from its preference for uncertain nodes. For example, with CI for Cora, it selects labels for which the current prediction was incorrect 56% of the time, compared to only 26% of the time with the default CLASSSEEK. Arguably,

this should result in more new information available for learning. However, this effect comes at a cost: the greater uncertainty means that the selection in step 9 of Figure 1, based on the *predicted* label, is much more likely to fail. Indeed, (with CI for Cora) step 9 succeeds in selecting a node with the desired *true* label 75% of the time with CLASSSEEK, but only 44% of the time with $+Margin$. This difference, combined with a difference in average node degree (for V^K, 9.2 vs. 2.6 with $+Margin$) leads to significant wins for CLASSSEEK vs. $+Margin$ in many cases.

VIII. CONCLUSIONS AND FUTURE WORK

Performing effective LBC requires the provision of a sufficient number of labeled nodes, which can be very expensive to obtain. Using active learning to reduce the total cost of label acquisition has great potential, but using the conclusions from prior studies was hampered by conflicting results and a lack of comparisons that considered important recent developments for LBC. In response, this paper produced the first study of active learning for LBC that considered recent improvements to semi-supervised learning for LBC, the use of neighbor attributes as features, and effective corrections for relational bias to prevent "flooding." Using such methods is important: Section VII discussed how the introduction of stronger learning likely affects previous findings, and we also saw that using neighbor attributes (with RI) can produce very different behavior. We also introduced two new active learning methods, and showed that one of them, CLASSSEEK, produced many statistically significant gains, while the existing methods frequently did no better than random guessing. CLASSSEEK also involves negligible computational cost, simply degree-based sampling and comparing predicted labels against a target label.

Our results need to be confirmed with additional datasets and learning algorithms. We will also study methods for computing the class distribution estimate D^* needed by CLASSSEEK, and the impact of errors in this estimate. Fortunately, estimates of D^* should be easier to obtain than labeled examples, since the information can be approximated and/or anonymous, and need not be based on a very large sample. For instance, preliminary results suggest that estimating D^* with 150 random samples generally yields accuracy on our datasets that is within 1% of CLASSSEEK's accuracy with the true class distribution.

CLASSSEEK usually won vs. the other AL methods, even though it selected nodes whose true labels wound up correcting prediction errors only 25-35% of the time. This "error selection rate" (a measure of informativeness) is on par with that of RANDOM, but is much lower than that of methods such as UNCRTN. Overall, CLASSSEEK selects nodes that are highly influential (due to degree sampling) and highly representative (due to its label balancing), but are only average for informativeness and distributedness. Future work should consider whether modifications to CLASSSEEK could improve upon the latter two objectives without markedly harming the former.

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] X. Shi, Y. Li, and P. Yu, "Collective prediction with latent graphs," in *Proc. of CIKM*, 2011, pp. 1127–1136.
[4] B. Gallagher, H. Tong, T. Eliassi-Rad, and C. Faloutsos, "Using ghost edges for classification in sparsely labeled networks," in *Proc. of KDD*, 2008, pp. 256–264.
[5] R. Xiang and J. Neville, "Pseudolikelihood EM for within-network relational learning," in *Proc. of ICDM*, 2008, pp. 1103–1108.
[6] L. K. McDowell and D. W. Aha, "Semi-supervised collective classification via hybrid label regularization," in *Proc. of ICML*, 2012, pp. 975–982.
[7] 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.
[8] M. Bilgic, L. Mihalkova, and L. Getoor, "Active learning for networked data," in *Proc. of ICML*, 2010, pp. 79–86.
[9] A. Kuwadekar and J. Neville, "Relational active learning for joint collective classification models," in *Proc. of ICML*, 2011, pp. 385–392.
[10] M. Ji and J. Han, "A variance minimization criterion to active learning on graphs," in *Proc. of AISTATS*, 2012, pp. 556–564.
[11] T. Kajdanowicz, R. Michalski, K. Musial, and P. Kazienko, "Active learning and inference method for within network classification," in *Proc. of ASONAM*. ACM, 2013, pp. 1299–1306.
[12] 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.
[13] M. Bilgic and L. Getoor, "Effective label acquisition for collective classification," in *Proc. of KDD*, 2008, pp. 43–51.
[14] 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.
[15] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *Proc. of KDD*, 2004, pp. 593–598.
[16] M. Saar-Tsechansky and F. Provost, "Active sampling for class probability estimation and ranking," *Machine learning*, vol. 54, no. 2, pp. 153–178, 2004.
[17] D. D. Lewis and W. A. Gale, "A sequential algorithm for training text classifiers," in *Proc. of SIGIR*, 1994, pp. 3–12.
[18] N. Roy and A. Mccallum, "Toward optimal active learning through sampling estimation of error reduction," in *Proc. of ICML*, 2001, pp. 441–448.
[19] J. Zhu, H. Wang, T. Yao, and B. K. Tsou, "Active learning with sampling by uncertainty and density for word sense disambiguation and text classification," in *Proc. of ACL*, 2008, pp. 1137–1144.
[20] C. Moore, X. Yan, Y. Zhu, J.-B. Rouquier, and T. Lane, "Active learning for node classification in assortative and disassortative networks," in *Proc. of KDD*, 2011, pp. 841–849.
[21] Z. Yang, J. Tang, B. Xu, and C. Xing, "Active learning for networked data based on non-progressive diffusion model," in *Proc. of WSDM*. ACM, 2014, pp. 363–372.
[22] J. J. Pfeiffer, III, J. Neville, and P. N. Bennett, "Active exploration in networks: Using probabilistic relationships for learning and inference," in *Proc. of CIKM*, 2014, pp. 639–648.
[23] F. Lin and W. W. Cohen, "Semi-supervised classification of network data using very few labels," in *Proc. of ASONAM*, 2010, pp. 192–199.
[24] M. Sharma and M. Bilgic, "Most-surely vs. least-surely uncertain," in *Proc. of ICDM*, 2013, pp. 667–676.
[25] 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.
[26] J. Neville and D. Jensen, "Relational dependency networks," *Journal of Machine Learning Research*, vol. 8, pp. 653–692, 2007.