A Hidden Treasure? Evaluating and Extending Latent Methods for Link-based Classification

Aaron Fleming, Luke K. McDowell, and Zane Markel
Dept. Computer Science, U.S. Naval Academy
572M Holloway Rd, Annapolis, MD 21402
(410) 293-6800
amfleming15@gmail.com,lmcdowel@usna.edu,zanemarkel@gmail.com

***Abstract*—Many information tasks involve objects that are explicitly or implicitly connected in a network, such as webpages connected by hyperlinks or people linked by "friendships" in a social network. Research on *link-based classification* (LBC) has studied how to leverage these connections to improve classification accuracy. This research broadly falls into two groups. First, there are methods that use the original attributes and/or links of the network, via a link-aware supervised classifier or via a non-learning method based on label propagation or random walks. Second, there are recent methods that first compute a set of *latent* features or links that summarize the network, then use a (hopefully simpler) supervised classifier or label propagation method. Some work has claimed that the latent methods can improve accuracy, but has not adequately compared with the best non-latent methods. In response, this paper provides the first substantial comparison between these two groups. We find that certain non-latent methods typically provide the best overall accuracy, but that latent methods can be competitive when a network is densely-labeled or when the attributes are not very informative. Moreover, we introduce two novel combinations of these methods that in some cases substantially increase 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 the label for node v, a classifier might use v's attributes along with features based on the proportion of v's neighbors that have a positive class label. This will require some kind of iterative *collective inference*, since many such labels will initially be unknown [3]–[5]. In some cases, simpler "relational-only" methods may yield high accuracy. These methods use only the known labels of the graph (ignoring attributes), and make predictions based on label propagation [6], [7] or random walks [8].

Link-based classification has been actively studied for over a decade [1], [2], [9], and continues to attract significant interest in the machine learning [10], [11], data mining [12]–[14], and knowledge management communities [15]. Applications of LBC have included, for instance, fraud detection [16], document topic prediction [4], biological discovery [3], and movie revenue estimation [17]. Many methods for LBC have been studied, and a number of comparisons made [4], [5], [17], [18]. One common property of these methods has been that they generally use the attributes and links of the network "as-is." Some methods have incorporated useful pre-processing, such as attribute dimensionality reduction [10] or "co-citation" link insertion [7]. These steps, however, have not been essential; if omitted the described methods would still function and produce reasonable results.

In contrast, a number of recent methods construct new "latent" features or links that are essential to a later inference step. Some of these latent computations use both the attributes and links of the original network. For instance, Latent Network Propagation (LNP) [15] transforms the original network by adding latent links between every pair of nodes, where each link is weighted based on a variety of factors such as attribute similarity or proximity in the original network. A label propagation algorithm then makes final label predictions for each node. Alternatively, other methods use only the network link structure. For instance, the "SocDim" method uses spectral clustering [19] or modularity maximization [20] to generate a new set of latent features for each node. A supervised (but link-unaware) classifier can then use the latent features to predict labels for all "unknown" nodes in the network.

In theory, these latent methods transform the original network into a form that is more amenable to analysis. For instance, spectral clustering with SocDim may identify structure in the graph that is helpful for prediction, enabling increased classification accuracy but without the need for constructing relational features (such as the "proportion" feature discussed above) or for iterative collective inference. Likewise, to assign the final edge weights, LNP uses quadratic programming to more strongly link together nodes that are likely to share the same class label. This enables a simple label propagation algorithm to predict labels, again avoiding the need for relational features or more complex collective inference.

The accuracy of the latent methods, however, has not been adequately compared against non-latent methods. For instance, Tang & Liu [20] compare SocDim against several baselines, but not against the best LBC methods and only using datasets with weakly-predictive attributes. Shi et al. [15] compare LNP against a popular LBC algorithm (ICA). However, they used a weak variant of ICA; we recently showed that its accuracy can be greatly improved with more effective semi-supervised learning [11] and through a new use of neighbor attributes as model features [18]. Thus, none of the latent methods have been evaluated against the best and most recent non-latent methods, and their relative performance remains unknown.

Our contributions are as follows. First, to aid understanding we present a new taxonomy that categorizes most existing LBC methods based on two key factors involving (a) the extent to which they use latent transformations and (b) the type of inference used. Second, we present the first substantial comparison between LBC methods based on latent features/links vs. those that use the original network attributes and links, including the best and most recent enhancements for both groups. Third, we show that the latent methods can perform effective LBC, but that the best non-latent methods consistently yield higher accuracy, especially when the original network is sparsely-labeled. Finally, inspired by our observations of the latent methods, we propose two new methods that combine the latent methods with an existing non-latent method. We show that, usually, a non-latent method continues to obtain the best performance, but that these new hybrid methods can sometimes increase accuracy when the network is very sparsely labeled.

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. If we exclude (for now) latent methods, 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`."

Table I characterizes (non-latent) 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 [7] (abbreviated in this paper as "wvRN") repeatedly averages the predicted label distributions of a node's neighbors; this performs surprisingly well for some datasets. Alternatively, MultiRankWalk [8] uses multiple random walks, each starting from a known label, to predict labels for the unknown nodes.

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].

TABLE I. Types of models, based on the kinds of features used. Table II provides more detail, including models with latent features or links.

Model	Self attrs.	Neigh. attrs.	Neigh. labels
AttrsOnly	✓		
RelatOnly			✓
CI	✓		✓
RI	✓	✓	
RCI	✓	✓	✓

We focus on ICA, a simple, popular, and effective algorithm [4], [10], [11]. 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.

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.

III. Latent Methods for LBC

This section describes how latent methods for LBC arose partially as a reaction to difficulties with applying existing methods to sparsely-labeled networks, then explains some of the primary approaches to generating latent features or links.

A. The Challenge of Sparsely-Labeled Networks

Of the LBC models shown in Table I (including CI, RI, and RCI), CI has been by far the most popular. This is due in part to early work that showed that RI sometimes hurt accuracy compared to using attributes alone [1], and also to a influential study that found that CI generally outperformed RI and did at least as well as RCI [3]. In addition, neighbor attributes (as used by RI or RCI) were generally considered to be incompatible with popular classifiers such as logistic regression and SVMs. Thus, CI was deemed more applicable and effective, so it was predominately used.

This approach worked well for tasks where learning was performed on a fully-labeled training graph, and inference was done on a separate, partially-labeled test graph [4], [17]. However, as researchers began to study the alternative, common case where only a single partially-labeled graph is available for both learning and inference [10], [12], [15], [21], problems with CI became more apparent. In particular, with CI a particular link in the graph can be directly used for learning only when *both* nodes attached to the link have known labels.[1] If, however, only 10% of the nodes are labeled, then perhaps only 1% of the links will satisfy this criteria.

Prior studies sought to address CI's limitations in several ways. Some tried using semi-supervised learning (SSL) to predict the missing labels [10]–[12]. Others sought to forgo learning altogether by using a non-learning, relational-only

[1] This enables the estimation of values such as "If v is labeled `Student`, what is the probability that a neighbor of v is labeled `Professor`?"

algorithm such as wvRN or MRW. In contrast, our own recent work [18] presented a new method that broadens the applicability of using neighbor attributes (which are known) instead of neighbor labels (which often are not); we showed that the resultant models (forms of RI and RCI) outperformed the non-learning methods as well as CI with SSL.

The next subsection describes how latent methods, in different ways, seek to address these same challenges.

B. Latent Feature and Latent Link Methods for LBC

Latent methods seek to improve LBC by adding either additional links or features to the original network. These additions may enable simpler inference and/or avoid the need for learning, changes which were partially inspired by the challenges of learning in sparsely-labeled networks [15], [21].

We first consider latent methods that use only the network link structure as inputs, ignoring all attributes and labels. One early approach to this idea was the "ghost edges" method of Gallagher et al. [21]. They use repeated random walks to quantify, for each unlabeled node, its proximity to every labeled node. Each such pair then gets a new link in the graph, with weight based on the measured proximity. They then use the new links to classify each node using either (a) wvRN or (b) a supervised classifier in conjunction with ICA. They report strong accuracy compared to a number of competitors, but only consider tasks where no attributes were available.

Other methods use the graph link structure not to create more links but to create latent features for each node. For instance, Tang & Liu [19] perform spectral clustering on the link structure to extract latent features, while Tang et al. [22] use a scalable k-means clustering of the links to produce latent features that are certain to be sparse (e.g., having few non-zero values). In each case, the new features are then used as features for a link-unaware supervised classifier, where learning uses only the labeled nodes and their new latent features.

The above methods all generate latent features or links in an unsupervised manner, e.g., ignoring the provided labels until learning or inference time. In contrast, Menon & Elkan [13] use an approach similar to that of Tang & Liu [20], but where the known labels influence the latent feature construction; they report mixed results for the impact on accuracy. Shi et al. [15] use the provided labels, and also the node attributes. Their goal is to construct new latent links so that the modified network has high homophily (i.e., nodes are very likely to link to other nodes with the same label), so that a simple inference procedure can then be used. In particular, they generate five fully-connected "latent graphs" where link weights are based on a quantity such as attribute similarity or proximity in the original graph. They then use quadratic programming to identify an optimal set of weights for combining these five graphs so that homophily among the known labels is maximized. Finally, they use a simple form of label propagation, similar to wvRN, to predict labels using the new links.

Other work uses matrix factorization in some form to produce new latent features [23]–[25]; these approaches may be supervised or unsupervised. They typically have used smaller datasets; scalability is a challenge [26] for some of them.

IV. Combining Latent and Non-Latent Methods

Above we have described LBC methods that use the original attributes and links, as well as others that create new latent features or links. Before directly comparing these two groups, this section first considers whether these two classes of methods could be profitably combined in novel ways.

Combining Latent Features with Collective Inference. A convenient property of latent features is that they, in theory, summarize a large amount of relevant information regarding the neighborhood of a node into a fixed number of latent features. Because the number of such features is fixed (unlike the varying number of links that each node has) a standard vector-based classifier can then make a prediction for each node, ignoring the links. Thus, there is no need for collective inference or iteration of any kind.

While prior work has shown that this method can improve accuracy over baselines such as wvRN, there remains the opportunity to take advantage of additional information. For instance, since methods based on, e.g., modularity maximization or spectral clustering all ignore the attribute information, it is natural to combine the resultant latent features with attribute-based features for classification. Tang & Liu [20] found that this could improve accuracy for a single blog-based dataset.

We propose, however, to use additional information. In particular, while Tang & Liu's approach uses "self attributes" and "self latent features", we also propose to use "neighbor labels" and/or "neighbor attributes" as features. For instance, if we use spectral clustering to generate latent features, then add self attributes and neighbor labels to the model (e.g., CI from Table I), then we call the new model "CI+SP."

While the latent and "neighbor-based" features both summarize information about a node's linked neighbors, we conjecture that they capture somewhat complementary information. Thus, using both may be helpful for classification. Adding neighbor labels will necessitate the use of collective inference, such as with ICA, but this is a fairly modest computational cost. Section VII considers whether this new combination of latent features with neighbor-based features is useful, or if using just one suffices for achieving maximal accuracy.

Combining Latent Links with Improved Label Propagation. As described above, LNP uses the original links and node attributes to generate a new, fully-connected network where homophily is (hopefully) high, so that a simple label propagation algorithm can be used to generate label predictions. Specifically, LNP uses the label propagation method of Zhu et al. [6] (herein referred to as LPP), which can also be viewed as a principled random walk in the network. Macskassy & Provost [7] previously found LPP to give almost identical results to wvRN. However, more recently Lin & Cohen [8] proposed a new method, MultiRankWalk (MRW), which is also a random walk-based method, but uses different initial conditions. They found found that MRW outperformed wvRN when the network was sparsely labeled (cf., [27]).

Thus, we propose a new method, LNP+MRW, that creates a new latent network like LNP, but uses MRW instead of LPP for the final step of inference. We expect this method to yield higher accuracy than LNP when the labels are very sparse, but to yield lower accuracy when the labels are more dense.

TABLE II. A TAXONOMY OF THE 15 METHODS STUDIED BY THIS WORK. CHECKMARKS INDICATE THAT THAT TYPE OF FEATURE IS USED BY THE METHOD TO MAKE THE FINAL LABEL PREDICTIONS. IN CONTRAST, "$\mathcal{I}$" SYMBOLS INDICATE THAT THAT TYPE OF FEATURE IS AN INPUT USED TO PRODUCE LATENT FEATURES OR LINKS. FOR INSTANCE, "ATTR+SP" USES THE LINK STRUCTURE OF THE GRAPH TO GENERATE (VIA SPECTRAL CLUSTERING) LATENT FEATURES; THESE FEATURES, ALONG WITH "SELF ATTRIBUTES" ARE LATER USED BY A "SINGLE PASS" CLASSIFIER (E.G., A LINK-UNAWARE CLASSIFIER SUCH AS SVM; SEE SECTION VI-C) TO MAKE FINAL PREDICTIONS.

Model	Self attrs.	Neigh. attrs.	Neigh. labels	Link structure	Latent feats.	Latent links	Inference method	Prior/original work (with sparse LBC)
Algorithms without Latent Features/Links								
ATTRSONLY	✓						Single pass	[10], [11], [18]
WVRN (weighted-vote relat. neighbor)			✓				Relax. labeling	[7]
LPP (label propagation)			✓				Label prop.	[6], [15]
MRW (MultiRankWalk)			✓				Random walks	[8]
CI (collective inference)	✓		✓				ICA/Gibbs/etc.	[10], [11], [15]
RI (relational inference)	✓	✓					Single pass	[18]
RCI (collective relational inference)	✓	✓	✓				ICA/Gibbs/etc.	[12], [18]
Algorithms with Latent Features/Links								
LNP (Latent Network Propagation)	$\mathcal{I}$		$\mathcal{I}$	$\mathcal{I}$		✓	Label prop.	[15]
LNP+MRW	$\mathcal{I}$		$\mathcal{I}$	$\mathcal{I}$		✓	Random walks	**None** (new)
MODMAX (modularity maximization)				$\mathcal{I}$	✓		Single pass	[20]
EDGE (edge clustering)				$\mathcal{I}$	✓		Single pass	[22]
SP (spectral clustering)				$\mathcal{I}$	✓		Single pass	[19]
ATTR+SP	✓			$\mathcal{I}$	✓		Single pass	[19]
CI+SP	✓		✓	$\mathcal{I}$	✓		ICA/Gibbs/etc.	**None** (new)
RCI+SP	✓	✓	✓	$\mathcal{I}$	✓		ICA/Gibbs/etc.	**None** (new)

V. A TAXONOMY OF LBC METHODS

For our purposes, most LBC methods can be categorized at a high level by answering two key questions. First, does the method use the original graph attributes and links, or does it generate and use latent features or links? Second, to make a final label prediction for each node, does it use a supervised classifier such as logistic regression, or does it use a non-learning method based on label propagation or random walks?

Table II summarizes key information about the 15 LBC methods that we evaluated in our experiments, organized according to these questions (see also the layout of Table IV). The top section of the table shows non-latent methods, while the bottom section shows latent methods. Within each section, methods that use label propagation or random walks are (mostly) shown first, followed by methods that use a supervised classifier. Methods that use a supervised classifier either apply it once to each node ("single pass") or use a collective inference method such as ICA or Gibbs sampling.

Among the non-latent methods, WVRN, LPP, and MRW are all non-learning, relational-only methods (see Section II) that use label propagation or random walks. CI is what most prior work means by methods for "collective inference" or "collective classification." Recently, we showed [18] that, when labels are sparse, using neighbor attributes (with RI) instead of neighbor labels (with CI) generally yielded higher accuracy, while using both (with RCI) was often best. Finally, ATTRSONLY uses only node attributes. ATTRSONLY, CI, RI, and RCI all use a supervised classifier, but ATTRSONLY is listed first in the table because of its status as a simple baseline.

We now consider latent methods. LNP is a latent link method, while LNP+MRW is our proposed improvement (see Section IV) that uses MRW (based on random walks) instead of LPP (label propagation). In contrast, MODMAX, EDGE, and SP all construct latent features, then generate final predictions using a supervised classifier. To construct the features, they use modularity maximization, edge clustering, or spectral clustering, respectively.

TABLE III. DATA SETS SUMMARY.

Characteristics	**Cora**	**CiteSeer**	**Gene**	**HepTH**
Total nodes	2708	3312	1103	2194
Total links	5278	4536	1672	9752
Class labels	7	6	2	7
% dominant class	16%	21%	56%	36%

Our results later show that SP yielded higher accuracy than EDGE or MODMAX. Thus, we use SP in our combinations of latent features with other information. ATTR+SP combines node attributes with SP, as done by Tang & Liu [20]. The last two methods, CI+SP and RCI+SP, are novel combinations of latent features with CI and RCI (see Section IV). We use CI because it has often been used in other work, and RCI because of its past superior performance as discussed above [18].

VI. EXPERIMENTAL METHOD

A. Datasets and Features

We used four commonly-used real datasets (see Table III):

Cora (cf., [4]) is a collection of machine learning papers and **Citeseer** (cf., [4]) is a collection of research papers. Attributes represent the presence of certain words, and links indicate citations. We mimic Bilgic et al. [10] by ignoring link direction, and also by using the 100 top attribute features after applying PCA to all nodes' attributes.

Gene (cf., [3]) describes the yeast genome at the protein level; links represent protein interactions. We mimic Xiang & Neville [12] and predict protein localization using four attributes: Phenotype, Class, Essential, and Chromosome. With logistic regression we binarized these, yielding 54 attributes.

HepTH is a set of journal articles on high-energy physics, as processed by McDowell et al. [5]; links represent citations. PCA was again used to produce the top 100 attribute features.

We focus on cases where the attributes are at least moderately predictive. Thus, we did not use previous datasets, e.g., based on Flickr and BlogCatalog [20], where this is not true; future work should study this other case more closely.

B. Classifiers and Regularization

In Table II all of the methods except the non-latent, relational-only methods (WVRN, LPP, MRW) and the LNP-based methods require learning a classifier to predict the label based on self attributes, neighbor attributes, and/or latent features. By default, we use logistic regression (LR), because it usually outperformed other alternatives [4], [10], [18]. However, to faithfully replicate the SocDim technique as originally presented [20], [22], we used SVM instead for MODMAX, EDGE, and SP. The combinations (e.g., ATTR+SP, CI+SP) used LR. Where SVM was used, we also ran separate experiments with LR, which showed very similar results.

RCI and CI also require a classifier to predict based on neighbor labels. McDowell & Aha [11] found that Naive Bayes (NB) with "multiset" features was superior to LR with "proportion" features as used by Bilgic et al. [10]. 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 [11]. RI and RCI also require a method to leverage the attributes from a varying number of neighbors; we use the "MNAC" method [18].

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, a common L2-regularization with SVM, and a Dirichlet prior with NB's discrete features (for neighbor labels). With the latent features (e.g., with EDGE, SP, and variants), we also used cross-validation to select an appropriate number of latent features to retain for the classification. For the most realistic scenario, we repeat cross-validations for each "trial", using the full network but only the specific "known" labels designated for that trial. This is in contrast to most earlier work, which either did not use cross-validation [15], [19], [20] or used some kind of "one-time" cross-validation that selected to keep, e.g., 500 latent features for a specific dataset regardless of the actual known labels available to each trial [22].

To ensure proper implementation, we obtained and used (after adding cross-validation) code for LNP, MODMAX, EDGE, and SP directly from their authors. We also validated our implementations by comparing vs. the results of Bilgic et al. [10] and Shi et al. [15]. We observed similar results, though they are not directly comparable because of different network samplings. Our accuracy values with LNP are smaller than those reported by Shi et al. because they included "known" nodes when computing overall accuracy.

C. Learning and Collective Inference

CI, RI, RCI, CI+SP, and RCI+SP require further learning and inference choices; we chose strategies that performed well in prior studies [11]. For learning, we use the SSL-ONCE strategy: first, learn the classifiers using the attributes and the known labels. Next, run inference to predict labels for every "unknown" node. Finally, learn new classifiers, using the attributes, known labels, and newly predicted labels. McDowell & Aha [11] found that these choices performed well overall and had consistent accuracy gains compared to not using SSL.

In Table II, methods marked "single pass" perform inference by applying the learned classifier once, using the attributes and features available for each node. Methods marked with "ICA/Gibbs/etc." instead use collective inference; we use 10 iterations of ICA (see Section II). We chose ICA because it has often performed well [4], [10], [11]. Label propagation (with LPP and LNP) used an "alpha" value of 0.01, as done by Shi et al. [15]; different values had minimal effect.

VII. RESULTS

We report accuracy averaged over 20 trials. For each, 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 [11], [15], [21], e.g., $d \leq 10\%$.

A. Overview of the Results

Before presenting detailed results, we first summarize our main findings. We find that, when the network is densely labeled (e.g. $d \geq 40\%$), most LBC methods performed well. Of the latent methods, SP usually performs best for this densely-labeled situation, and even obtains the best accuracy of any method for HepTH when $d = 80\%$. However, when the network is only sparsely labeled ($d \leq 10\%$), accuracy varies widely between the different methods, and using RCI or RI yields substantially higher accuracy than all other methods. The other methods also have somewhat erratic behavior. For instance, SP performs relatively well for Cora, but very poorly for Citeseer. In contrast, RCI is a consistent performer; it had maximal or near maximal accuracy in every case.

For the sparsely-labeled case, there was no clear winner among the latent methods. For the latent link methods, SP has higher accuracy than EDGE and MODMAX (averaged across all four datasets), but EDGE sometimes performed just as well or a little better. Likewise, SP sometimes performs a little better than LNP (a latent feature method), but LNP yields much higher accuracy than SP on two datasets.

Overall, these results show, for the first time, that the latent methods can be competitive in some limited cases (especially when the label density is high), but that non-latent methods like RCI and RI consistently provide the best overall accuracy. The next sub-sections provide more details and also examine the results with the new methods introduced in Section IV.

B. Comparing Existing Latent Methods

Table IV shows the average accuracy of each method on the four datasets for cases where $d \leq 10\%$. We first compare the latent methods to each other:

Result 1: Spectral clustering (SP) **outperforms** EDGE **and** MODMAX **on average, but is not always best.** Among the latent link methods, SP has higher accuracy than EDGE

TABLE IV. ACCURACY RESULTS FOR EACH DATASET FOR VARYING VALUES OF THE LABEL DENSITY d. THE DOUBLE VERTICAL BARS DIVIDE THE TABLE INTO THREE SECTIONS: ONE FOR NON-LATENT METHODS, AND TWO DIFFERENT SECTIONS FOR THE LATENT METHODS. WITHIN EACH HORIZONTAL SECTION, WE SHOW THE MAXIMAL VALUE FOR EACH ROW IN BOLD.

	Algorithms without Latent Features/Links							Algorithms with Latent Features or Links						
	ATTRS-ONLY	Label prop/walks			Sup. classifier			Label prop/walks		Supervised classifier				
		wvRN	LPP	MRW	CI	RI	RCI	LNP	LNP+MRW	EDGE	SP	ATTR+SP	CI+SP	RCI+SP
A.) Cora														
$d = 1\%$	37.1	41.2	54.6	58.4	57.2	67.8	**71.3**	50.5	53.3	40.9	**56.0**	48.1	64.2	**70.3**
$d = 3\%$	54.1	63.4	64.2	67.2	75.1	78.7	**80.0**	66.5	65.8	56.3	**69.7**	64.1	76.6	**79.2**
$d = 5\%$	59.5	72.3	67.8	70.3	78.3	80.5	**81.6**	72.6	69.5	61.6	**73.1**	69.5	78.9	**80.5**
$d = 10\%$	65.8	77.9	71.7	73.8	81.3	81.8	**83.2**	**77.6**	71.3	65.8	75.6	75.2	81.2	**82.2**
Average	54.1	63.7	64.6	67.4	73.0	77.2	**79.0**	66.8	65.0	56.1	**68.6**	64.2	75.2	**78.1**
B.) Citeseer														
$d = 1\%$	40.7	31.5	38.2	38.1	52.5	57.9	**59.1**	36.3	**39.0**	26.8	29.6	42.4	46.9	**54.9**
$d = 3\%$	55.8	47.9	45.1	46.2	64.2	66.7	**67.2**	**55.0**	52.3	38.0	40.8	55.6	60.4	**65.2**
$d = 5\%$	61.8	52.6	48.2	49.5	67.7	69.4	**69.7**	**63.3**	59.0	42.9	47.0	59.0	64.3	**68.0**
$d = 10\%$	66.7	55.8	52.2	53.7	69.6	71.3	**71.4**	**68.1**	62.3	45.8	52.3	61.2	68.1	**70.6**
Average	56.3	47.0	45.9	46.9	63.5	66.3	**66.9**	**55.7**	53.2	38.4	42.4	54.5	59.9	**64.7**
C.) Gene														
$d = 1\%$	59.8	55.7	61.4	62.6	60.9	65.9	**67.1**	57.9	**60.8**	53.0	52.4	57.2	60.8	**66.5**
$d = 3\%$	65.5	62.6	65.5	66.4	66.7	71.6	**73.6**	65.5	**65.6**	57.9	54.7	65.1	66.6	**73.7**
$d = 5\%$	68.4	66.6	68.7	69.9	70.6	75.7	**77.6**	**69.2**	68.9	59.5	56.0	66.2	70.9	**77.4**
$d = 10\%$	71.6	71.6	70.8	72.4	74.2	78.0	**79.4**	**73.6**	73.2	65.0	62.4	70.2	74.4	**79.3**
Average	66.3	64.1	66.6	67.8	68.1	72.8	**74.4**	66.5	**67.1**	58.8	56.4	64.7	68.2	**74.2**
D.) HepTH														
$d = 1\%$	35.2	38.6	39.0	34.6	26.4	**39.9**	38.1	**38.7**	28.6	36.0	33.0	32.0	26.7	**38.6**
$d = 3\%$	42.3	43.7	45.1	40.5	30.0	**49.9**	48.1	**43.5**	34.4	42.7	42.2	43.3	38.8	**47.4**
$d = 5\%$	44.9	46.6	47.3	43.1	35.9	**52.4**	49.8	36.6	36.3	44.8	**45.0**	46.8	41.6	**50.6**
$d = 10\%$	49.3	51.0	50.5	45.9	43.0	**55.4**	53.0	36.3	38.1	47.8	**51.1**	51.0	45.5	**52.7**
Average	42.9	45.0	45.5	41.0	33.8	**49.4**	47.3	38.8	34.4	**42.8**	**42.8**	43.3	38.2	**47.3**

(averaged across all four datasets), but EDGE performs just as well on HepTH and a little better than SP on Gene. Results with MODMAX are not shown due to lack of space; compared to SP and EDGE it never had the highest accuracy and almost always had the lowest. The lower accuracy of MODMAX is consistent with prior results [19], [22], but SP and EDGE have not, to our knowledge, been previously compared.

Result 2: On average, LNP **produces higher accuracy than** SP**, but** SP **outperforms** LNP **for some datasets.** SP (a latent link method) sometimes performs a little better than LNP (a latent feature method), but LNP yields much higher accuracy on two datasets, which gives SP an advantage in overall average accuracy. Specifically, LNP yields much higher accuracy with Citeseer and Gene, two datasets where (a) the attributes are fairly predictive and (b) the homophily is moderate. In contrast, LNP is a little worse than SP on Cora (where high homophily means that relational-only methods like SP can do very well) and on HepTH (where adding attributes to any model is not very helpful).

Result 3: Adding attributes to latent features boosts accuracy, but still yields low relative accuracy. The latent feature methods use only the network links and known labels; attribute values are ignored. If, after computing the latent features, we add attributes to the model (e.g., with ATTR+SP), then accuracy improves vs. SP for all datasets except Cora. Surprisingly, however, the average accuracy obtained is still less than a method that uses only the attributes (ATTRSONLY), except for HepTH where there is a very small improvement.

In theory, adding informative latent features (e.g., from SP) to the provided node attributes should improve accuracy. Indeed, Tang & Liu [19] find that this method provides a small but consistent improvement compared to ATTRSONLY. However, they studied this effect only for a single dataset, and used a task where nodes may have more than one true label. Moreover, they used a fixed number of latent features for each dataset, rather than using cross-validation (CV) to select an appropriate number based on the actual known labels for each trial. In our case, we suspect that, when sparsely labeled, the CV (and subsequent learning) has difficulty with the larger set of features from ATTR+SP (typically, the CV retains at least hundreds of latent features). Indeed, when the network is more densely-labeled, ATTR+SP does better (e.g., with Cora, accuracy at $d = 20 - 40\%$ is about 2% better than SP or ATTRSONLY). Future work should study these effects more.

C. Latent Methods vs. Non-latent Methods

We now contrast the best of the non-latent methods (left side of Table IV) with the best of the latent methods:

Result 4: RCI **and** RI **consistently yield the highest accuracy compared to other methods.** Table IV shows that either RCI or RI almost always provides the highest accuracy, across all four datasets. Specifically, RCI is best for Cora, Citeseer, and Gene, while RI is best for HepTH. RCI's gains are often substantial; e.g., on Cora its average accuracy of 79.0% is more than 10% higher than the best relational-only baseline (MRW at 67.4%) and the best latent method (SP at 68.6%).

Thus, for our datasets using RCI or RI is best when the network is sparsely-labeled. Figure 1 shows the results for a full range of label densities ($1\% \leq d \leq 80\%$). For clarity, we focus on just five methods: the best overall method (RCI), the

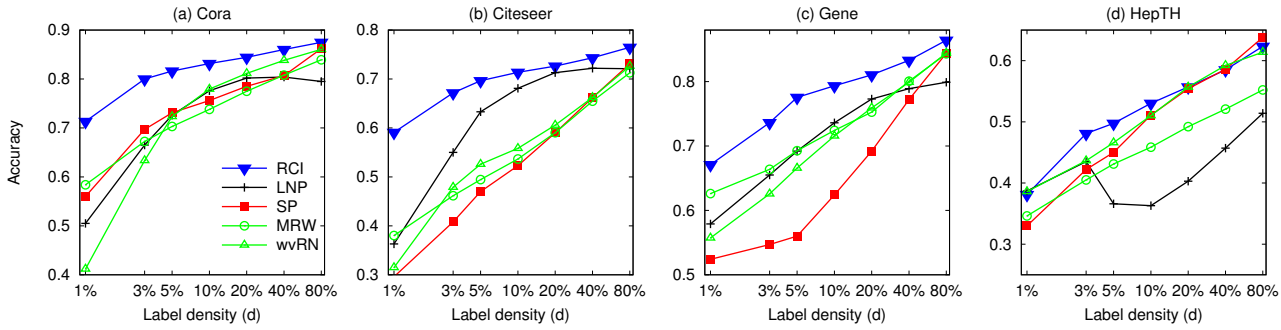

Fig. 1. Average accuracy for five representative LBC methods. LNP and SP are latent methods, while the others are non-latent methods.

best latent methods (LNP and SP), and the two most common non-latent relational-only methods (wvRN and MRW).

Figure 1 shows that RCI continues to perform very well across all values of d. Excluding RCI, the other methods usually have competitive accuracy when when the network is very densely labeled, but sometimes have erratic behavior. For instance, SP performs well on Cora (for all d) and HepTH (for $d \geq 5\%$), even having the best overall accuracy for HepTH when $d = 80\%$. With Gene, however, SP is the worst method (often by far) for $d < 80\%$ and is also usually at the bottom for Citeseer. LNP also has somewhat erratic behavior, yielding mostly high accuracy on most datasets, but having very poor accuracy on HepTH (in the latter case, a few trials appear to suffer from "flooding" of the network [4] that leads to poor accuracy overall).

Result 5: Unlike with prior studies, CI almost always outperforms LNP and other latent methods. Shi et al. [15] compared CI (specifically, a version of semi-supervised ICA) to their new LNP method, and found that CI yielded very poor accuracy. However, we later showed how to substantially improve CI's accuracy with the use of a hybrid classifier and with improved SSL [11]. We also argued that Shi et al.'s conclusions (regarding the superiority of LNP) would likely change if this stronger form of CI was used, but did not specifically compare against LNP.

Table IV shows that our argument was correct: CI outperforms LNP by a significant margin on all datasets except HepTH; the same is true regarding CI vs. the latent link methods (Edge and SP). Even on HepTH, Figure 1 shows that LNP does not perform very well when $d \geq 5\%$, and indeed it lags the accuracy of CI when $d \geq 10\%$ (CI results not shown). Thus, while Shi et al.'s comparison of LNP vs. CI was appropriate at that time, their conclusion that "traditional collective classifiers and their semi-supervised modifications do not perform well" [15] was too pessimistic regarding the relative accuracy of CI vs. latent methods like LNP.

D. Results with the New Latent/Non-Latent Combinations

We now examine the two new kinds of methods for combining latent and non-latent methods from Section IV.

Result 6: Adding latent features to a traditional collective classifier can improve accuracy when the network is very sparsely labeled. In Table IV, adding latent features to CI (with CI+SP) improves accuracy for two of the datasets (Cora and HepTH) when $d < 10\%$. The gains are substantial in some cases, e.g., by up to 7% for Cora and up to 9% for HepTH, with an average gain of 2.2% for Cora and 4.4% for HepTH for $1\% \leq d \leq 10\%$. For both datasets, even better accuracy could obtained by using RCI instead (where adding the SP-based features does not help). Nonetheless, these results show that using the new CI+SP can sometimes improve over CI, and this may be the best approach for a dataset where using neighbor attributes (as with RI or RCI) is not desirable, perhaps due to missing attribute values. Naturally, CI+SP will be most helpful for datasets where SP also works relatively well on its own. Thus, in Table IV CI+SP did not work well for Citeseer and Gene, which were also the two datasets where SP fared poorly compared to other methods like MRW or LNP.

Result 7: When the network is very sparsely labeled, LNP+MRW yields small but mostly consistent gains vs. LNP. We expected that using MRW with LNP instead of LPP would yield higher accuracy when d was small. Table IV confirms this effect: when $d = 1\%$, LNP+MRW increases accuracy by about 3% for Cora, Citeseer, and Gene. Surprisingly, however, LNP+MRW generally decreased accuracy for higher values of d, even when d is still quite small (e.g. $3 - 10\%$). Two things may explain the limited scope of these gains. First, LNP+MRW is replacing LPP (not wvRN, for which previous comparisons were known) with MRW, and while MRW does have higher accuracy than LPP in Table IV for Cora, Citeseer, and Gene, its gains are small and usually not as large as vs. wvRN. Second, LNP transforms the original network into a fully connected graph (with weighted links). The advantages of MRW vs. LPP do not necessarily carry over to this much denser network, and indeed these results suggest that MRW's advantages in this case are present when d is very small (1%), but not usually for higher values of d.

VIII. Related Work

Most prior comparisons of LBC methods only considered cases with fully-labeled training graphs [3]–[5], [17]. A few comparisons [7], [11], [12] have instead examined the "within network" task that we focus on, with a single partially-labeled graph, but none considered the latent methods of Table II.

A few "within network" studies have compared latent methods to each other or to non-latent methods, but left significant questions unanswered. For instance, prior work showed that usually SP outperformed ModMax [19] and Edge outperformed ModMax [22], but did not compare SP

to EDGE or compare either of these to LNP, CI, RI, or RCI. Shi et al. [15] compared LNP to CI, but only to a weak CI variant (see Section VII-C). Menon & Elkan [13] compare three latent methods, including MODMAX, but find conflicting results and do not compare to better latent methods (like SP) or to strong non-latent methods like CI and RCI.

For LBC, cross-validation to select regularization parameters can have a large impact on accuracy, but is challenging, especially for sparsely-labeled networks. Menon & Elkan [13] use such a process, but find that over-fitting is still a problem for some of their datasets. This may also be an issue with the poor results for ATTR+SP (see Section VII-B); future work should consider alternative regularization strategies.

Macskassy & Provost compared WVRN vs. LPP and report "nearly identical" results (though they describe some differences in node ranking). Later papers (e.g., [8]) took this to mean that the methods are equivalent and can be interchanged freely. We showed, surprisingly, that their accuracy can differ markedly, especially if labels are very sparse, for which case LPP outperformed WVRN on all of our datasets.

IX. CONCLUSION

Link-based classification is an important task, for which the most common methods involve computing relational features and performing collective inference. Recently, a number of competing methods based on constructing latent features or links have been shown to have some promise. To date, however, comparisons between non-latent methods and the new latent methods have been absent or inadequate.

This paper presents the first results that compare state of the art non-latent methods with recently proposed latent methods. We found that most methods can yield high accuracy when the network is densely labeled, but that accuracy varies widely when labels are more sparse. Specifically, we showed that latent link methods like LNP outperformed, on average, the best latent feature methods (e.g., SP), but that none of these latent approaches was consistently better than others. However, we found that RCI (and often RI and CI) almost always outperformed the latent methods when the labels were sparse. Moreover, RCI was very consistent, producing maximal or near maximal accuracy in all cases. Thus, for our datasets a non-latent method, RCI, was generally the best choice. We also proposed, however, two new methods that combine latent links or features with an existing non-latent method. We found that they could sometime improve accuracy vs. other methods when the network was very sparsely labeled.

To aid understanding, we also presented a new taxonomy of LBC methods. In this taxonomy, the type of method depends upon two distinctions: (a) whether the method uses the original attributes and links vs. using latent features or links, and (b) for inference, whether the methods uses label propagation or random walks vs. using a supervised classifier. Future work should evaluate other methods that fit within this taxonomy (e.g., [21]), as well as others that do not cleanly fit (e.g., [14]).

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

[6] 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.

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

[8] F. Lin and W. W. Cohen, "Semi-supervised classification of network data using very few labels," in *Proc. of ASONAM*, 2010, pp. 192–199.

[9] B. Taskar, P. Abbeel, and D. Koller, "Discriminative probalistic models for relational data," in *Proc. of UAI*, 2002, pp. 485–492.

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

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

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

[13] A. Menon and C. Elkan, "Predicting labels for dyadic data," *Data Mining and Knowledge Discovery*, vol. 21, no. 2, pp. 327–343, 2010.

[14] G. Namata, S. Kok, and L. Getoor, "Collective graph identification," in *Proc. of KDD*, 2011, pp. 87–95.

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

[16] J. Neville, Ö. Simsek, D. Jensen, J. Komoroske, K. Palmer, and H. G. Goldberg, "Using relational knowledge discovery to prevent securities fraud," in *Proc. of KDD*, 2005, pp. 449–458.

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

[18] 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.

[19] L. Tang and H. Liu, "Leveraging social media networks for classification," *Data Mining and Knowledge Discovery*, pp. 1–32, 2011.

[20] ——, "Relational learning via latent social dimensions," in *Proc. of KDD*, 2009, pp. 817–826.

[21] 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.

[22] L. Tang, X. Wang, and H. Liu, "Scalable learning of collective behavior," *IEEE Transactions on Knowledge and Data Engineering*, 2011.

[23] S. Zhu, K. Yu, Y. Chi, and Y. Gong, "Combining content and link for classification using matrix factorization," in *Proc. of SIGIR*. ACM, 2007, pp. 487–494.

[24] P. Hoff, "Multiplicative latent factor models for description and prediction of social networks," *Computational & Mathematical Organization Theory*, vol. 15, no. 4, pp. 261–272, 2009.

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

[26] A. Menon and C. Elkan, "Link prediction via matrix factorization," *Machine Learning and Knowledge Discovery in Databases*, pp. 437–452, 2011.

[27] R. Crane and L. McDowell, "Investigating markov logic networks for collective classification." in *Proc. of ICAART*, 2012, pp. 5–15.