Multi-label Collective Classification using Adaptive Neighborhoods

Tanwistha Saha, Huzefa Rangwala and Carlotta Domeniconi
Department of Computer Science
George Mason University
Fairfax, Virginia, USA
tsaha@gmu.edu, {rangwala,carlotta}@cs.gmu.edu

***Abstract*—Multi-label learning in graph-based relational data has gained popularity in recent years due to the increasingly complex structures of real world applications. *Collective Classification* deals with the simultaneous classification of neighboring instances in relational data, until a convergence criterion is reached. The rationale behind collective classification stems from the fact that an entity in a network (or relational data) is most likely influenced by the neighboring entities, and can be classified accordingly, based on the class assignment of the neighbors. Although extensive work has been done on collective classification of single labeled data, the domain of multi-labeled relational data has not been sufficiently explored. In this paper, we propose a neighborhood ranking method for multi-label classification, which can be further used in the Multi-label Collective Classification framework. We test our methods on real world datasets and also discuss the relevance of our approach for other multi-labeled relational data. Our experimental results show that the use of ranking in neighborhood selection for collective classification improves the performance of the classifier.**

***Keywords*-Multi-label; co-authorship networks; collective classification**

I. Introduction

Multi-label learning in graph-based relational data has gained popularity in recent years due to the increasingly complex structures of real world applications. Examples of relational data include social networks, biological protein-protein interaction (PPI) networks, and research collaboration networks. In social networks, different entities (people) are associated with multiple groups and interests. Traditional multi-label classification algorithms (except a few [1], [2], [3]) treat every node independently and assume independence among different labels. As such, these methods perform poorly on multi-labeled relational datasets.

Given a network, three possible correlations are relevant while determining the labels of nodes [4]: (i) the correlation between the label of a node and the observed attributes of the node, (ii) the correlation between the label of a node and the observed attributes and labels of neighboring nodes, and (iii) the correlation between a known label of a node and the unobserved labels of neighboring nodes. Collective classification approaches jointly classify a set of related (linked) nodes by exploiting the above underlying correlations [5]. Most collective classification methods are applicable to single-labeled relational data [6], [7], [8], [9]. Recently, a method was proposed for the multi-label collective classification problem [10]. However, none of these approaches discuss how to differentiate between the influence of labels of "closely-related" neighbors from that of "unrelated" neighbors.

We approach the problem of multi-labeled collective classification by combining three different types of information. Firstly, we use the information that is associated with a given node in the form of attributes or features. Secondly, we use information associated with a given node in the form of "other" labels. Finally, within the collective classification paradigm, we use the label information provided by the node's neighborhood. Our primary contribution is realizing that, for a given node, not every neighboring node has the same influence when predicting its multiple labels. As such, we develop a method to *rank* the neighboring nodes using the pairwise interaction information between them, coupled with the association of neighboring nodes with different labels. We then propose a method for *pruning* the neighborhood using the *rank* values, which is incorporated within the standard collective classification algorithms.

We evaluate the performance of our approach on a collaboration network of authors, who have published papers in different research areas. For our problem, we know the labels (i.e., the research areas) of some of the nodes (i.e., authors); the task is to predict the labels of the remaining unlabeled nodes (i.e., the most likely research areas for authors). Our experimental results on co-authorship network data (http://www.informatik.uni-trier.de/~ley/db/) show that pruning the neighborhoods during collective classification improves the prediction performance.

II. Related Work

Collective classification is a semi-supervised method, first proposed almost ten years ago [7], [6]. State-of-the-art collective classification methods on single labeled datasets are described in a comprehensive survey [4]. Conventional collective classification approaches focus on classifying single labeled datasets, and can be broadly categorized into *local* and *global* methods. Local methods (e.g., logistic regression, naive Bayes) classify each test instance using relational and attribute features, and later update the labels and relational features through an iterative process until convergence [6], [7]. McDowell *et al.* [9] proposed a *cautious* approach to exploit intermediate relational data which can be noisy. This approach computes relational features by filtering links between the different pairs of nodes. Macskassy *et al.* have developed

a toolkit that combines different components e.g., use of any local classifier with any collective inference algorithm [8]. Global methods optimize an objective function across the entire relational dataset using topological and node-level information [11]. The classifier uses both attributes and relational features for inference. Examples of methods used for inference include Loopy Belief Propagation (LBP) and Mean Field Relaxation Labelling (MF) [4]. Gallagher *et. al.* have proposed another approach [12] which attempts to find features that are independent of node labels and analyzed the effects of label-independent features on within-network classification. Besides the recently proposed work of Kong *et al.* [10], not much has been done towards the development of multi-labeled collective classification algorithms.

Multi-label learning focuses on the prediction of multiple labels for each test instance simultaneously. Several algorithms have been proposed in this area (see the survey in [13]). Zhang *et al.* developed a *k*-nearest neighbors based method for multi-label classification, which treats each label independently, and then computes the probability of assigning a particular label [2]. Another method proposed by Zhang uses *label specific* features for classifying instances [1]. Zhang *et al.* also proposed a Bayesian network based approach to capture the conditional dependencies between labels as well as the feature set [3]. A recently developed algorithm simultaneously ranked and classified objects within the DBLP citation data [14]. This algorithm was designed for a heterogeneous network of different types of objects (nodes), and computes the *within-class* ranking of these objects. In this paper, we discuss an approach for multi-labeled collective classification within a homogeneous network. In our approach, the use of word *rank* as a measure of *influence* is strictly to impose a pruning strategy, for determining those neighboring nodes that would affect the prediction of a label. This is different from the approaches that study influence of a node within a relational network [15], e.g., to determine influencing nodes for a viral marketing campaign.

III. Methodology

Let graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ represent a relational dataset $D = (\mathcal{X}, \mathcal{Y})$, where $\mathcal{X} = \{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_n\}$ is the set of node attributes and $\mathcal{Y} = \{\mathbf{y}_1, \mathbf{y}_2, \cdots, \mathbf{y}_n\}$ is the set of corresponding label vectors for the $|\mathcal{V}| = n$ nodes in the network. Let K be the number of classes, denoted by $k = 1, 2, \cdots, K$, to which the nodes can belong. Given the set L of training nodes, the training data $D_L = (\mathcal{X}_L, \mathcal{Y}_L)$, the set U of unlabeled test nodes and the set of node attributes $\mathcal{X}_U$ for the test nodes in U; the goal is to predict the label vector $\mathbf{y}_u = (y_{u1}, \cdots, y_{uk}, \cdots, y_{uK})$ of each test node $u \in U$, where $y_{uk} \in \{0, 1\}$. For convenience, we denote $\hat{\mathbf{Y}}_U = \{\hat{\mathbf{y}}_1, \cdots, \hat{\mathbf{y}}_u, \cdots, \hat{\mathbf{y}}_{|U|}\}$ as the set of label vectors predicted by our models, and $\mathbf{Y}_U = \{\mathbf{y}_1, \cdots, \mathbf{y}_u, \cdots, \mathbf{y}_{|U|}\}$ as the set of true label vectors for all the test nodes in U. This problem is similar to collective classification in a single labeled dataset, except that, in this case, we have to predict multiple labels for each of the test nodes. This paper aims to exploit the information hidden in the interactions between entities by using properties of graphs. In addition to capturing information from the topology of the graph, we also capture the *association* between a node and its label. We *prune* the neighborhood of a test node based on the "*ranks*" of the training nodes, so that, while predicting the label of the test node, we only trust the label information propagated from the *influential* neighbors.

A. RankNN Algorithm

We propose RankNN (Algorithm 1) which estimates the labels of test nodes using a naive Bayes approach. Assuming that a node's label is influenced by that of its neighbors, a node's posterior label assignment might change based on the beliefs shared by its neighborhood. Given a set L of training nodes and K labels, we compute the ranking (importance) for each of the labels. The rank r_k for label k is given by (line 3 in Algorithm 1),

$$r_k = \frac{|L_k|}{|L|} \tag{1}$$

where L_k is the set of training instances with label k. Using Equation 1, the label rank vector $R = (r_1, r_2, \cdots, r_k, \cdots, r_K)$ of all the K labels is determined. We then compute the rank of a labeled node l as follows (line 5 in Algorithm 1),

$$rank_l = \sum_{j \in N_l} C_{lj} A_{lj} + \sum_{k=1}^{K} M_{lk} r_k \tag{2}$$

where N_l is the set of labeled neighbors of node l. $A_{n \times n}$ is the weighted adjacency matrix representing the number of interactions between any pair of nodes in the network. We use the weighted adjacency matrix for training nodes only. $C_{n \times n}$ is the pairwise cosine similarity measure between the attributes of the entities. $M_{n \times K}$ is a matrix representing the association of a node, say l, with a label, say k. Equation 2 can be explained as: (i) The first term captures the level of interactivity of an individual. This is measured by the similarity the author has with his neighbors (captured by the cosine similarity matrix $C_{n \times n}$), weighted by the number of interactions (co-authored papers) the author has had with them. (ii) The second term measures the reputation of the individual in terms of his association with different groups (or labels), and the author's contribution towards those groups (captured by $M_{n \times K}$). (iii) We define the *rank of a label* as the frequency of entities in the training data that has that corresponding label (e.g., r_k in Equation (1) is rank of label k). A highly reputed group will tend to have a large number of members, and hence will be ranked higher.

For our DBLP collaboration network, the association between an author and a research area is determined by the number of publications he/she has in the conferences under that research track. If the author is a prolific researcher but does not publish too many papers in a highly ranked conference, then the second term will weigh down his/her rank. Similarly, if the author has several publications in

Algorithm 1 RankNN

Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, weighted adjacency matrix $A_{n\times n}$, similarity matrix $C_{n\times n}$, node and label association matrix $M_{n\times K}$, total number of classes K, probability threshold θ, number of iterations T, training percentage η for sampling the data into training set L and test set U

Output: Set of label vectors of nodes in test set U i.e. $\hat{\mathbf{Y}}_U$

1: **for** $t = 1$ to T **do**
2: Sample training set L and test set U based on training percentage η
3: Compute ranks of labels $\forall k \in K$ according to Equation (1)
4: **for** each node l in L **do**
5: Compute the rank of node l according to Equation (2)
6: Set threshold δ as median of the rank values
7: **end for**
8: **for** each node u in U **do**
9: Compute prior probability $\hat{\mathbf{\Pi}}_u$ from Equation (3)
10: Find all *influential* neighbors N_u^i (of node u) s.t. $\forall j \in N_u^i, rank_j \geq \delta$
11: **for** $k = 1$ to K **do**
12: Compute likelihood $\Pr\left[Z_u = k | y_{uk} = 1\right]$ from Equation (4)
13: Calculate the posterior probability of $y_{uk} = 1$ from Equation (5)
14: **end for**
15: **end for**
16: **for** each node u in U **do**
17: Normalize the posterior probabilities of $\mathbf{y}_u$ across all K
18: **if** $\Pr\left[y_{uk} = 1 | Z_u = k\right] > \theta$ **then**
19: Set $\hat{y}_{uk} = 1$
20: **else**
21: Set $\hat{y}_{uk} = 0$
22: **end if**
23: **end for**
24: **end for**

reputed conferences, which are not related to the research area (label) we are investigating, then the second term will be under weighted. The rank of the conference in an unrelated research area will be globally low, as determined by Equation (1). On the other hand, if an author has published few papers in highly ranked conferences, and has collaborated with too many researchers, then the first term will decrease the author's rank compared to the rank of other researchers who have many co-authors and have published in highly ranked conferences.

Next, we compute the prior probabilities $\hat{\mathbf{\Pi}}_u$ for each unlabeled test node u from the information contained in the local neighborhood. $\hat{\mathbf{\Pi}}_u = (\hat{\pi}_{u1}, \hat{\pi}_{u2}, \cdots, \hat{\pi}_{uK})$ is a K dimensional vector of probabilities, where each element of the vector corresponds to the probability of a label. Let N_u be the set of labeled neighbors of an unlabeled node u. Let L_j be the set of labels of the labeled neighbor $j \in N_u$. Let N_{uk} be the set of labeled neighbors of node u that has label k. Then the prior probability of label k for node u is given by (line 9 in Algorithm 1):

$$\hat{\pi}_{uk} = \Pr\left[y_{uk} = 1\right] = \frac{|N_{uk}|}{|\cup_{j\in N_u} L_j|} \tag{3}$$

However, if the unlabeled node u does not have labeled neighbors, then we assign equal probability $\frac{1}{K}$ to all the labels of that node. This is dependent on the percentage of available labeled data, i.e., if we have very few labeled nodes then there is a high chance that an unlabeled test node will *not* have any labeled neighbors. We assess the performance of our algorithms by varying the sampling ratios.

Algorithm RankNN estimates the posterior probability of each label for each test node. Let N_u^i be the set of influential neighbors (of a test node u) having rank values above a threshold δ (set dynamically as median of the calculated rank values, although other values of δ are also possible). Let Z_u be a random variable defined over the distribution of all possible data labels, and defined as follows: $Z_u = k$ if k is the most frequent label within the neighborhood of u. Let L_j^i be the set of labels of the labeled influential neighbor $j \in N_u^i$, where N_u^i is the set of influential neighbors of node u. Let N_{uk}^i be the set of influential nodes in the neighborhood of node u that have label k (includes node u based on the assumption that node u has label k), then the probability that u will also have label k depends on the following likelihood (line 12 in Algorithm 1):

$$\Pr\left[Z_u = k | y_{uk} = 1\right] = \frac{|N_{uk}^i|}{|\cup_{j\in N_u^i} L_j^i|} \tag{4}$$

The posterior probability used for determining whether node u has label k, i.e., y_{uk} (line 13 in Algorithm 1) is:

$$\Pr\left[y_{uk} = 1 | Z_u = k\right] \propto \Pr\left[y_{uk} = 1\right] \times \Pr\left[Z_u = k | y_{uk} = 1\right] \tag{5}$$

Algorithm 1 describes the pseudo-code for RankNN. The computational complexity of RankNN is $O(|U|KT)$ where $|U|$ is the number of unlabeled nodes, K is the number of labels and T is the number of iterations ($|L| \leq |U|$).The two input parameters for RankNN are: (i) rank threshold δ and (ii) the probability threshold θ. The rank threshold δ, controls the number of influential neighbors that should be selected while determining the labels for a test node. The RankNN algorithm produces a probabilistic output for the different class labels. As such, to convert the probability scores into hard label assignments (0/1) we use the probability threshold parameter (θ).

B. ICML_Rank Algorithm

Multi-label Collective Classification methods [10] couple the attribute level information with two types of relational information: (i) dependencies that exist between the multiple labels assigned to a node (referred to as *intra-instance cross-label dependencies*) and (ii) the dependencies between the labels of neighboring node and the node in consideration. The *ICML* algorithm [10], as termed by the authors, is based on iterative classifications of multiple labels while treating each label individually. We improve this approach by incorporating our node ranking function within the multi-label collective classification framework. Instead of blindly considering all the neighbors, we *prune* the neighborhood of a test node based on the ranks of its neighbors while computing the relational features for that node (line 7 in Algorithm 2). We set the threshold δ dynamically as the median of the obtained rank values. We use LIBSVM [16] with a linear kernel for learning classifiers for each of the K labels. In ICML_Rank, a model is learned with training samples for each of the labels $k = 1, 2, \cdots K$. An initial set of labels for test nodes is predicted after feeding the test instances to the trained models. This initial label set for test nodes is used to recompute the ranks of *all* the nodes in the network. Inference of the labels for the test nodes is carried out through an iterative process. Relational features

Algorithm 2 ICML_Rank

Input: A graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, weighted adjacency matrix $A_{n\times n}$, similarity matrix $C_{n\times n}$, node and label association matrix $M_{n\times K}$, total number of classes K, number of iterations T, training percentage η for sampling the data into training set L and test set U

Output: Set of label vectors of nodes in test set U i.e. $\hat{\mathbf{Y}}_U$

for $t = 1$ to T **do**
  Training:
  Sample the dataset into training L and test U based on training percentage η
  for $k = 1$ to K **do**
    Compute the rank of label k from Equation (1)
    Compute the ranks of each node $l \in L$ from Equation (2)
    Set the threshold δ, such that nodes with ranks above δ are *influential*
    Find the relational features of each node $l \in L$ considering only influential neighbors
    Construct an extended training set $D_L^k = (\mathcal{X}_L^k, \mathcal{Y}_L^k)$ by combining attribute x_l of each node l with its relational features, to give x_l^k
    Train a local classifier using D_L^k
    Let $f^k = A(D_L^k)$ be the learned local model for label k
  end for
  Bootstrapping:
  for each node $u \in U$ **do**
    Estimate the label $\hat{\mathbf{y}}_u = (\hat{y}_{u1}, \hat{y}_{u2}, \cdots, \hat{y}_{uK})$ such that, $\hat{y}_{uk} = f^k(\mathbf{x}_u, \mathbf{0})$ using only node attributes and no relational features
  end for
  Iterative Inference:
  Compute the initial ranks of test nodes by averaging the ranks of labels predicted in the bootstrapping phase
  repeat
    Construct an extended test set $D_U^k = (\mathcal{X}_U^k, \hat{\mathbf{Y}}_U^k)$ (same method as in training) using the ranked influential neighbors
    Update the estimated values of $\hat{\mathbf{y}}_u$ for $\mathbf{y}_u$ on each test node $u \in U$ as, $\hat{y}_{uk} = f^k(D_U^k)$
    Update the ranks of the labels (equation (1)) and ranks of *all* the nodes (equation (2)), based on this new label assignment $\hat{\mathbf{y}}_u$ of each node $u \in U$
  until convergence criteria OR maximum number of iterations
  Return the label vector $\hat{\mathbf{y}}_u$ for each test instance $u \in U$
end for

Table I
DESCRIPTION OF DATASETS.

Dataset	#Nodes	#Classes	#Degree/node	#Classes/node
DBLP_A	10334	7	7.1608	1.6205
DBLP_B	6379	6	5.6247	1.1130

are computed for the test nodes, and along with attribute features, are fed to the learned models to get a new set of predicted labels. This step is repeated until a maximum number of iterations is reached, or until a given accuracy value is achieved. Algorithm 2 gives the pseudo-code of ICML_Rank.

IV. EXPERIMENTAL SETUP

A. Datasets

For our experiments, we filtered the DBLP citation network data[1] and created two datasets which are denoted as: DBLP_A and DBLP_B. The authors in DBLP_A have published two or more papers from year 2000 to year 2010 in seven research areas identified by the conference names: Databases (SIGMOD, VLDB, EDBT, ICDE, PODS), Data Mining (KDD, SDM, ICDM), Artificial Intelligence (AAAI, IJCAI, AAMAS, UAI), Software Engineering (ICSOFT), Computer Vision (CVPR), Information Retrieval (SIGIR, ECIR), Machine Learning (ICML, ECML). The DBLP_B dataset consists of authors who have published papers in six different research areas: Algorithms & Theory (FOCS, STOC, SODA, COLT), Natural Language Processing (ACL, ANLP, COLING), Bioinformatics (ISMB, RECOMB), Networking (SIGCOMM, MOBICOM, INFOCOM), Operating Systems (SOSP, OSDI), Distributed & Parallel Computing (PODC, ICDCS). Table I provides key statistics for both the datasets. A graph $\mathcal{G}$ represents the DBLP network data containing labeled and unlabeled nodes. Each node in the graph represents an author; node attributes correspond to the titles of the papers that an author has published (attributes are obtained as *tf-idf* measure and are collectively treated as *a document vector* for the author node; relational features are computed following Kong *et. al* [10]). Labels of each node (or author) correspond to the research areas the author has been working on. Since each author may be interested in more than one research area, authors can have multiple labels. A link between a pair of nodes exists if the corresponding authors have co-authored at least one paper. The weights on the edges of the adjacency matrix represent the number of papers they have co-authored. We also have a weight matrix that captures the pairwise cosine similarity between the titles of the papers that a given pair of authors have co-authored. Using the structure of the collaboration network, given the titles of the published papers and multiple labels of authors in the training set, the goal is to predict the multiple labels for authors in the test set.

[1]downloaded from http://arnetminer.org/

B. Validation Protocol

To create an unbiased validation set, we remove all papers (titles) from the training set that are published by authors within the test set. As such, the authors in the training set do not have details about some of the papers that they have published. This leads to a modification in the individual labels for nodes within the training set. The test nodes do not possess any information that is shared with the training nodes. Most of our empirical evaluations are performed on this set, referred to as the "filtered" dataset. In previous work, this filtering process was not done [17], [7], [10], [3]. As such, the test and train instances would share the co-authored publications. To understand the nature of biased evaluation, we compare the performance of our algorithms on the original dataset, referred to as "unfiltered".

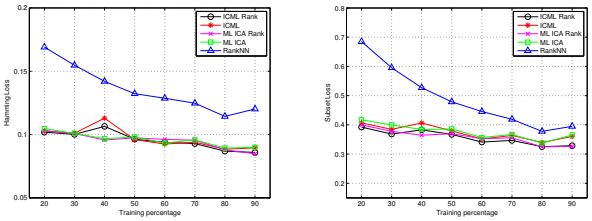
C. Evaluation Metrics

We use the following multi-label classification metrics [10] for evaluation. (i) **Hamming Loss (HL):** measures the fraction of incorrectly predicted labels, averaged across the set U of test nodes. (ii) **Subset Loss (SL):** evaluates if the predicted label vector is *exactly* identical to the true label vector (macro-label). (iii) **Micro-F1 score (MI–F1):** measures the harmonic mean of precision and recall across individual micro-labels, averaged across set U of test nodes. and (iv) **Macro-F1 score (MA–F1):** computes the average of the F1 measure on the predictions of the macro-labels.

V. RESULTS

We assess the performance of ICML_Rank and RankNN with varying parameters across the four multi-label evaluation metrics. Unless otherwise stated, the default parameter

Figure 1. Performance vs training percentage (η) for DBLP_A data.



(a) Hamming Loss (b) Subset Loss

for ICML_Rank is a pruning threshold ($\delta = 50\%$), and for RankNN (to convert the posterior probabilities into hard 0/1 label assignments) is $\theta = 0.25$. The performance of our algorithm is compared against two previously developed approaches (discussed in [10]): (i) **Iterative Classification of Multiple Labels (ICML):** Our ICML_Rank algorithm is inspired by this approach. ICML does not prune the neighborhood of a node while computing its relational features. (ii) **Multi Label Iterative Classification (ML_ICA):** This algorithm is similar to ICML except that it does not use the intra-instance cross label dependencies. Following a similar approach as in ICML_Rank, we incorporated our neighborhood pruning technique within the ML_ICA framework; we call the resulting approach **ML_ICA_Rank**.

Table II
PREDICTION PERFORMANCE (↑ MEANS HIGHER THE BETTER, ↓ MEANS LOWER THE BETTER) WITH FILTERED & UNFILTERED LABEL SETS FOR DBLP_A & DBLP_B DATASETS ($\theta = 0.25$ FOR RANKNN).

Labels	Method	HL ↓	SL ↓	MI-F1 ↑	MA-F1 ↑
Filtered	RankNN	0.1543	0.5946	0.5698	0.4851
	ML_ICA	0.1012	0.4023	0.7465	0.6278
	ML_ICA_Rank	**0.1001**	0.3847	**0.7529**	0.6377
	ICML	0.1040	0.3963	0.7451	0.6345
	ICML_Rank	0.1023	**0.3826**	0.7518	**0.6399**
Unfiltered	RankNN	0.1503	0.5672	0.5905	0.5083
	ML_ICA	0.0957	0.3266	0.7788	0.6596
	ML_ICA_Rank	**0.0966**	0.3182	0.7785	0.6638
	ICML	0.0960	0.3169	0.7809	0.6617
	ICML_Rank	0.0965	**0.3107**	**0.7810**	**0.6661**
DBLP_B					
Filtered	RankNN	0.1096	0.4199	0.7074	0.6859
	ML_ICA	**0.07951**	0.2630	**0.8037**	0.7474
	ML_ICA_Rank	0.0806	**0.2543**	0.8028	**0.7475**
	ICML	0.1213	0.3598	0.7025	0.6212
	ICML_Rank	0.1170	0.3451	0.7148	0.6409
Unfiltered	RankNN	0.1102	0.4287	0.7034	0.6797
	ML_ICA	0.0571	0.1657	0.8397	0.7598
	ML_ICA_Rank	0.0584	0.1609	0.8377	0.7605
	ICML	**0.0570**	0.1608	0.8411	0.7608
	ICML_Rank	0.0588	**0.1560**	**0.8479**	**0.7618**

A. Filtered Validation Set

Table II reports the average of 10 runs for HL, SL, MI–F1 and MA–F1 for DBLP_A "filtered" and "unfiltered" sets. The results for DBLP_B will be available as a supplementary file. For all the algorithms, we observe that the "unfiltered" set produces lower loss and higher F1 scores in comparison to the "filtered" set. This was expected, because the "unfiltered" set is biased and uses information available from the training nodes directly in the test nodes. Henceforth, we report and discuss results only for the unbiased "filtered" validation set.

Table III
PERFORMANCE WITH DIFFERENT RANK THRESHOLDS (δ).

δ	Method	HL ↓	SL ↓	MI-F1 ↑	MA-F1 ↑
DBLP_A					
0%	RankNN	0.1539	0.6044	0.5663	0.4699
	ICML_Rank	0.0968	0.3727	0.7548	**0.6449**
	ML_ICA_Rank	0.0972	0.3780	0.7612	0.6428
10%	RankNN	0.1545	0.5999	0.5664	0.4900
	ICML_Rank	**0.0948**	**0.3588**	**0.7619**	0.6399
	ML_ICA_Rank	0.0989	0.3615	0.7609	0.6378
25%	RankNN	0.1543	0.5971	0.5676	0.4890
	ICML_Rank	0.1094	0.3921	0.7366	0.6369
	ML_ICA_Rank	0.0971	0.3706	0.7619	0.6414
50%	RankNN	0.1543	0.5946	0.5698	0.4851
	ICML_Rank	0.1023	0.3826	0.7518	0.6399
	ML_ICA_Rank	0.1001	0.3847	0.7529	0.6377
75%	RankNN	0.1572	0.5961	0.5631	0.4657
	ICML_Rank	0.1024	0.3859	0.7508	0.6295
	ML_ICA_Rank	0.1013	0.3935	0.7486	0.6288
DBLP_B					
0%	RankNN	0.1086	0.4198	0.7077	0.6895
	ICML_Rank	0.1143	0.3335	0.7225	0.6641
	ML_ICA_Rank	0.0762	0.2386	0.8145	0.7612
10%	RankNN	0.1104	0.4269	0.7030	0.6842
	ICML_Rank	0.1082	0.3231	0.7361	0.6740
	ML_ICA_Rank	0.0803	0.2509	0.8046	0.7516
25%	RankNN	0.1103	0.4237	0.7056	0.6797
	ICML_Rank	0.1052	0.3181	0.7447	0.6920
	ML_ICA_Rank	**0.0756**	**0.2336**	**0.8147**	**0.7637**
50%	RankNN	0.1096	0.4199	0.7074	0.6859
	ICML_Rank	0.1170	0.3451	0.7148	0.6409
	ML_ICA_Rank	0.0806	0.2543	0.8028	0.7475
75%	RankNN	0.1104	0.4134	0.7075	0.6820
	ICML_Rank	0.1282	0.3704	0.6870	0.6046
	ML_ICA_Rank	0.0835	0.2745	0.7929	0.7248

B. Rank Threshold (δ)

RankNN and ICML_Rank use a dynamically selected threshold for the rank value, which determines the *influential* neighbors to be trusted while determining the label of a test node through collective classification. Table III shows the multi-label classification performance with varying δ parameter values for RankNN and ICML_Rank across the two DBLP datasets. A threshold (δ) of 25% signifies that 25% of the labeled nodes within the neighborhood of the unlabeled test node are pruned or removed. The ICML_Rank algorithm with a δ value of 0% does not prune any neighboring nodes and is identical to the ICML algorithm.

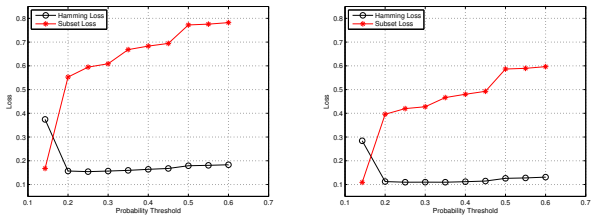
C. Comparative Performance

From Tables II and III we observe that ICML_Rank (ML_ICA_Rank) outperforms ICML (ML_ICA) across the DBLP datasets. For DBLP_A, ICML_Rank and ML_ICA_Rank shows the best performance. However, for DBLP_B, ML_ICA shows better performance in comparison to the ICML_Rank algorithm. We performed pairwise t-test on the results of ICML and ICML_Rank for DBLP_A (ML_ICA and ML_ICA_Rank for DBLP_B) and noted (at confidence level of 90%) *p-values* of 0.02 and 0.03, respectively. The ML_ICA algorithm does not use the dependency information across multiple assigned labels. Since the research areas (or labels) in DBLP_B are highly uncorrelated, discarding this dependency information in the relational features boosts ML_ICA's performance.

D. Sampling Ratio

To evaluate the sensitivity of our algorithms with regards to the training set size, we performed experiments that vary

Figure 2. Varying θ for RankNN.



(a) Loss vs θ for DBLP_A (b) Loss vs θ for DBLP_B

the training percentage from 20% to 90%. Figure 1 shows the variation for different evaluation metrics with respect to the training percentage (η) using the filtered DBLP_A set. The ICML_Rank algorithm performs better than the other methods even when the training percentage is very low, i.e., 20%. This shows the robustness of the algorithm, especially when the training data size is small.

E. RankNN Probability Threshold (θ)

The RankNN algorithm produces a probabilistic output for the different class labels. We use the probability threshold (θ) to convert the probability scores into a hard label assignments. In Figure 2 we show the hamming loss and subset loss for varying θ values. When we set θ to small values, we assign several more labels per node; this results in a higher hamming loss. However, setting θ to a larger value will result in fewer assignments of labels, thereby leading to a higher subset loss. From Figure 2 we observe that setting the value of θ between 0.2 and 0.45 seems to be optimal.

F. Computation Time

Table IV reports the computation time of different algorithms on 30% training/70% testing data for the DBLP datasets, and shows the advantage of the RankNN approach. Experiments were performed on a Intel Core 2 Duo, 3GHz, 4GB RAM desktop.

Table IV
COMPUTATION TIME FOR ALGORITHMS.

DBLP_A		DBLP_B	
Method	Time (seconds)	Method	Time (seconds)
RankNN	**46.97**	RankNN	**37.54**
ML_ICA	2054.46	ML_ICA	802.52
ML_ICA_Rank	2159.58	ML_ICA_Rank	912.88
ICML	2319.07	ICML	953.81
ICML_Rank	2693.40	ICML_Rank	1193.72

VI. CONCLUSION.

We have developed a novel and simple algorithm for computing the ranks of nodes within a relational network. We have incorporated our ranking technique within a multi-label collective classification framework to select the influential neighbors in an adaptive manner. We have tested our methods on a real world dataset, and reported promising results in comparison to state-of-the-art multi-label collective classification algorithms. We have shown that our methods ICML_Rank and ML_ICA_Rank perform well compared to the baseline algorithms ICML and ML_ICA, respectively. We also provided a useful insight towards creating an unbiased validation set for collective classification algorithms.

REFERENCES

[1] M. Zhang, "Lift: Multi-label learning with label-specific features," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

[2] M. Zhang and Z. Zhou, "Ml-knn: A lazy learning approach to multi-label learning," *Pattern Recognition*, vol. 40, 2007.

[3] M. Zhang and K. Zhang, "Multi-label learning by exploiting label dependency," in *ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2010, pp. 999–1008.

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

[5] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2004, pp. 593–598.

[6] L. Getoor, "Link-based classification," *Advanced methods for knowledge discovery from complex data*, pp. 189–207, 2005.

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

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

[9] L. McDowell, K. Gupta, and D. Aha, "Cautious inference in collective classification," in *National CONFERENCE ON ARTIFICIAL INTELLIGENCE*, vol. 22, no. 1, 2007, p. 596.

[10] X. Kong, X. Shi, and S. Philip, "Multi-label collective classification," in *SIAM International Conference on Data Mining (SDM)*. SIAM, 2011, pp. 618–629.

[11] B. Taskar, P. Abbeel, and D. Koller, "Discriminative probabilistic models for relational data," in *Eighteenth Conference on Uncertainty in Artificial Intelligence (UAI02)*, 2002.

[12] B. Gallagher and T. Eliassi-Rad, "Leveraging label-independent features for classification in sparsely labeled networks: An empirical study," *Advances in Social Network Mining and Analysis*, pp. 1–19, 2010.

[13] G. Tsoumakas and I. Katakis, "Multi-label classification," *International Journal of Data Warehousing & Mining*, vol. 3, no. 3, pp. 1–13, 2007.

[14] M. Ji, J. Han, and M. Danilevsky, "Ranking-based classification of heterogeneous information networks," in *17th ACM SIGKDD international conference on Knowledge discovery and data mining*, 2011, pp. 1298–1306.

[15] A. Goyal, F. Bonchi, and L. Lakshmanan, "A data-based approach to social influence maximization," vol. 5, no. 1. VLDB Endowment, 2011, pp. 73–84.

[16] C.-C. Chang and C.-J. Lin, "LIBSVM: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology*, vol. 2, pp. 27:1–27:27, 2011.

[17] Q. Lu and L. Getoor, "Link-based classification," in *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, vol. 20, no. 2, 2003, pp. 496–503.