

# Cautious Inference in Collective Classification

Luke K. McDowell<sup>1</sup>, Kalyan Moy Gupta<sup>2</sup>, and David W. Aha<sup>3</sup>

<sup>1</sup>Dept. of Computer Science; U.S. Naval Academy; Annapolis, MD 21402

<sup>2</sup>Knexus Research Corp.; Springfield, VA 22153

<sup>3</sup>Navy Center for Applied Research in Artificial Intelligence;

Naval Research Laboratory (Code 5514); Washington, DC 20375

lmcowell@usna.edu, kalyan.gupta@knexusresearch.com, david.aha@nrl.navy.mil

## Abstract

Collective classification can significantly improve accuracy by exploiting relationships among instances. Although several collective inference procedures have been reported, they have not been thoroughly evaluated for their commonalities and differences. We introduce novel generalizations of three existing algorithms that allow such algorithmic and empirical comparisons. Our generalizations permit us to examine how *cautiously* or *aggressively* each algorithm exploits intermediate relational data, which can be noisy. We conjecture that cautious approaches that identify and preferentially exploit the more reliable intermediate data should outperform aggressive approaches. We explain why caution is useful and introduce three parameters to control the degree of caution. An empirical evaluation of collective classification algorithms, using two base classifiers on three data sets, supports our conjecture.

## Introduction

Classification is the task of assigning one or more class labels to an unlabeled instance. Many supervised learning algorithms induce classifiers (e.g., that induce Bayesian networks, decision trees, neural networks, rules).

An underlying assumption of traditional learning methods is that the instances are independent of each other. However, instances in many classification tasks are often implicitly or explicitly related, such as when assigning topics to web pages. Hyperlinked web pages are more likely to share common class labels than non-linked pages, and this factor should be considered when classifying them. Such *auto-correlation* (correlation of class labels among interrelated instances) has been observed in a wide variety of data (Jensen and Neville 2002), including situations where the relationships are implicit. For example, email messages between two people are likely to share topics.

*Collective classification* is a methodology that simultaneously classifies related instances. It can increase classification accuracies over non-collective methods when instances are interrelated (Neville and Jensen 2000; Taskar, Abbeel, and Koller 2002; Lu and Getoor 2003). A number

of algorithms have been used for such collective inference, including relaxation labeling (Chakrabarti, Dom, and Indyk 1998), iterative convergence techniques (Neville and Jensen 2000; Lu and Getoor 2003), belief propagation (Taskar *et al.* 2002), and Gibbs sampling (Jensen, Neville, and Gallagher 2004).

In this paper, we observe that all collective inference algorithms exploit relational features based on uncertain (and thus noisy) class labels, which is potentially problematic. However, they differ in how (if at all) they manage such uncertainty while exploiting the relational features. We call algorithms *cautious* if they seek to explicitly identify and preferentially exploit the more certain relational information, and otherwise call them *aggressive*. On this basis, we develop novel generalizations of three existing algorithms. In particular, we introduce three parameters to control the degree of caution used by collective inference algorithms, and conjecture that *cautious algorithms should attain higher classification accuracies compared to their more aggressive variants*. Our investigations confirm that, indeed, the most cautious algorithms perform best, and that caution enables a simple iterative algorithm to outperform Gibbs sampling, a popular yet much more computationally expensive choice.

To date, collective inference algorithms have not been thoroughly evaluated for their commonalities and differences. Sen and Getoor (2006) did evaluate three algorithms on synthetic data using a single classifier, and found the Iterative Classification Algorithm (ICA) to be the most reliable. In this paper, we generalize two popular families of algorithms (including ICA), and evaluate them using two base classifiers on three real world datasets.

Our research is complementary to that of Jensen *et al.* (2004). They use a single collective algorithm to examine the impact that different types of relational links in the model have on accuracy. We instead vary the collective inference algorithm, focusing on how noisy intermediate class labels are exploited in the algorithm. This effect has only been partially addressed (Neville and Jensen 2000); we discuss this in more detail after presenting our results.

In the next section, we provide background on collective classification. We then introduce our generalization of collective classification algorithms and explain the parameters for their degree of caution in exploiting relational information. Finally, we present our experimental evaluation and conclude with future work.

## Collective Classification

In some classification tasks, the unlabeled instances can be implicitly or explicitly related. For example, consider classifying a university web page as belonging to a faculty member or a student when many such pages may interlink. Standard classifiers ignore such relations or links and would classify a web page by considering only the features derived from its contents (e.g., words it contains). Classification accuracy can be increased by adding features derived from the content of related instances (e.g., words from the hyperlinked web pages). Even greater accuracy increases can occur when the class label(s) of the related web pages are used to derive relevant *relational* features (Jensen *et al.* 2004). However, some or all of the class labels of the related web pages are initially unknown and need to be *estimated* or *inferred* to bootstrap the classification process. For instance, initial estimates of class labels can be obtained using content features only. Next, these estimates could be used to compute the values of the relational features and reclassify the instances. This iterative process of inferring and reclassifying could be repeated, possibly resulting in higher accuracy. However, since some labels will be incorrect, there is also the potential for relational features to *harm* performance. This is the motivation for cautiously exploiting such features, as we examine in the next section.

*Collective classification* algorithms simultaneously classify interrelated instances using estimated or inferred relational features. They have three primary characteristics:

- *Related instances*: The instances to be classified (e.g., web pages, emails) are explicitly or implicitly related.
- *Classifier*: To classify an instance  $i$  (e.g., a webpage), the base classifier uses *non-relational* features (e.g., the words in page  $i$ ) and *relational* features (e.g., the most common class label among other pages linked to  $i$ ). Many classifiers have been used for this purpose, including Naïve Bayes (Jensen *et al.* 2004), Markov networks (Taskar *et al.* 2002), case-based classifiers (McDowell, Gupta, and Aha 2007), and logistic regression (Lu and Getoor 2003). However, results with different classifiers have seldom been compared.
- *Collective inference*: An inference algorithm is used to update the class labels (or conditional probabilities), which are then used to recompute the relational feature values. As stated above, several inferencing techniques have been used (e.g., Gibbs sampling and belief propagation), but they have seldom been compared.

## Algorithms for Collective Inference

We introduce two generalizations of existing collective inference algorithms:

1. *Iterative Collective Classification* (ICC): ICC is a generalization of (i) the unnamed iterative algorithm of Neville and Jensen (2000), which is cautious, and (ii) the ICA algorithm of Lu and Getoor (2003), which is

```

ICC(Tr,Te,NR,R,n,C, Z) =
// Tr=Training data, Te=Test data, NR=non-relational features,
// R=rel. features, n=#iterations, C=classifier, Z=Bool. params
1 Tr.R.values←setRelFeatures(Tr,R,ZtreatAbsentAsUnknown)
2 M←induce_model(Tr,NR,R,C) // Train
3 Te.Labels←classify(Te,Tr,M,NR,∅,C) // Bootstrap
4 for J = 0 to n // Iterate
5 if (J == 0) && (ZfavorTrainLinks)
    Te.Labels←∅ //Set all to unknown; use train labels only
6 else if (ZfavorConfidentInstances) // Keep top K labels
    K = (J / n) * |Te|
    Te.Labels←commit_best_k (Te.Labels, K)
7 Te.R.values←setRelFeatures(Te∪Tr,R,ZtreatAbsentAsUnknown)
8 Te.Labels←classify(Te,Tr,M,NR,R,C) // Classify
9 return Te.Labels // return most likely class per test instance

```

**Figure 1.** Pseudocode for iterative collective classification.

aggressive. This generalization introduces parameters to control the propensity to use uncertain relational data.

2. *Parameterized Gibbs Sampling for Collective Classification* (PGCC) Like ICC, this is a generalization of Gibbs Sampling for collective classification that introduces a parameter to control how uncertain relational information is used.

We use a case-based classifier (k-NN) and a Bayesian classifier with these algorithms.

### Algorithm 1: Iterative Collective Classification

Figure 1 shows the pseudocode for the ICC algorithm, which includes three parameters to control its degree of caution. Before explaining the use of these parameters in detail, we describe the algorithm in its most aggressive mode (all three parameters are *false*) where it blindly uses all available relational information in its entirety.

**Aggressive ICC.** In step 1, the algorithm computes the values of all relational features for the fully labeled training set. In step 2, a model is learned using the training data. Step 3 is a bootstrapping step, where it predicts class labels for the test instances using only non-relational features. Steps 5-6 are ignored in the aggressive mode. In step 7, the algorithm updates the relational feature values based on its predictions. In step 8, it reclassifies the test set using all features. Steps 7 and 8 then repeat for  $n$  iterations. Lastly, step 9 returns the final set of class labels.

**Cautious ICC.** The aggressive mode of ICC can be tempered by setting three Boolean parameters:

**Parameter 1.**  $Z_{\text{favorConfidentInstances}}$  for *cautiously using intermediate instance labels*: In step 7 of ICC’s aggressive mode, the relational feature values are computed assuming that the assigned instance labels are all equally likely to be correct. We modify this aggressive behavior to be more cautious by only considering label assignments for which we are more confident. To do this, we set  $Z_{\text{favorConfidentInstances}}$  to *true*. With this setting, step 6 selects only the “best”  $K$  of

the current label assignments, “commits” those labels, and sets all other labels to *unknown*. Step 7 then computes the relational features using *only* the committed labels, and step 8 classifies using this information. Step 6 gradually increases the number of labels that are committed in each iteration (e.g., if  $n=10$ , then the iterations commit  $\{0\%, 10\%, 20\%, \dots, 100\%\}$  of the test set instances). Note that instances “committed” in one iteration are not necessarily committed again in the next.

Leaving some label assignments as *unknown* in step 6 impacts the relational feature value computation in step 7 in two ways. First, feature values are computed without using the *unknown* labels. Since this computation depends only on the most reliable label assignments, subsequent assignments should also be more reliable. Second, the computed value of some features will be *unknown* (e.g., when an instance links only to instances labeled *unknown*).

This approach requires determining the “best” K of the current label assignments. We adopt Neville and Jensen’s (2000) approach and use the probability of the most likely class for each instance as a confidence measure. In exploratory experiments, we found that alternative measures (e.g., probability difference of top two classes) produced similar results. This approach also requires that the classifier handle features with *unknown* values.

**Parameter 2**  $Z_{\text{favorTrainLinks}}$  for increasing caution by favoring linked training set instance labels over linked test set instance labels: A realistic collective classification scenario frequently involves a test set with links to training set instances (the “in-sample” task of Neville and Jensen (2005)). For instance, a new set of web pages may be published that link to pages with known labels. Such links between the test and training sets provide known instance labels (i.e., the most certain) for relational feature value computation. Therefore, we add another degree of caution by favoring the class labels of linked training instances over the most confident estimations of labels from the test set. To accomplish this, we set the  $Z_{\text{favorTrainLinks}}$  to *true*. With this setting, in the first iteration, the relational feature values are computed using *only* the links to instances in the training set (via setting all test set labels to *unknown* in step 5). The behavior of subsequent iterations depends on the value of  $Z_{\text{favorConfidentInstances}}$ .

**Parameter 3**  $Z_{\text{treatAbsentAsUnknown}}$  for cautiously handling absent links: Many machine learning algorithms assume that their input data have no missing values. However, relational features offer more complexity. In particular, consider a relational feature that is *true* when an instance has at least one outgoing link to an instance labeled “B”. What is the value of this feature when an instance  $i$  does not out-link to any other appropriate instances, regardless of their current labels? For example, an instance may only have incoming links or link only to instances not used in the current task. In such cases, the aggressive approach for using information entails setting the feature value to *false* – essentially treating the absence of any link data as a negative indicator for class “B.” Here, we consider a more cautious alternative. When  $Z_{\text{treatAbsentAsUnknown}}$  is *true*, we set

```

PGCC(Tr,Te,NR,R,n,C, Z) =
// Tr=Training data, Te=Test data, NR=non-relational features,
// R=rel. features, n=#iterations, C=classifier, Z=Bool. params
1 Tr.R.values←setRelFeatures(Tr,R, ZtreatAbsentAsUnknown)
2 M←induce_model(Tr,NR,R,C) // Train
3 Te.ClassProbs←classify(Te,Tr,M,NR,∅,C) // Bootstrap
4 for j=1 to n // Iterate
5 Te.Labels ←sampleDist(Te.ClassProbs) // Sample
6 Te.Stats←updateStats(Te.Stats,Te.Labels) // Take stats
7 Te.R.values←setRelFeatures(Te∪Tr,R,ZtreatAbsentAsUnknown)
8 Te.ClassProbs←classify(Te,Tr,M,NR,R,C) // Classify
9 Te.Labels←pickMostLikelyClass(Te.Stats)
10 return Te.Labels // return most likely class for each instance

```

**Figure 2.** Collective classification using Gibbs sampling.

the feature value to *unknown* for such instances. This allows the feature to distinguish (a) instances that have no appropriate links vs. (b) instances that do appropriately link to other instances, but where those instances’ labels are not “B.” The former case yields a feature value of *unknown*; the latter, *false*. If  $Z_{\text{treatAbsentAsUnknown}}$  were *false*, the feature value for both cases would be *false*.

**Relation to previous work:** With the parameters  $Z_{\text{favorConfidentInstances}}$  and  $Z_{\text{favorTrainLinks}}$  set to false, ICC reduces to the ICA algorithm of Lu and Getoor (2003) (aside from different interleaving of steps 7 and 8). Similarly, when only  $Z_{\text{favorConfidentInstances}}$  is set to true, ICC reduces to the iterative algorithm described in Neville and Jensen (2000). In their experiments, there were no effective links between the test and training sets, so  $Z_{\text{favorTrainLinks}}$  would have no effect. Regarding the parameter  $Z_{\text{treatAbsentAsUnknown}}$ , we found no discussion of this issue in the collective inference literature, although feature descriptions in several papers (e.g., Neville and Jensen 2000; Lu and Getoor 2003) suggest that a value of *false* would be (aggressively) assigned for the instances we described.

## Algorithm 2: Parameterized Gibbs Sampling for Collective Classification (PGCC)

Figure 2 summarizes how Gibbs sampling can be applied to collective inference. Steps 1-3 are identical to those in Figure 1, with the exception that the classifier must output distributions with the likelihood of each class. In step 5, within the loop, the algorithm probabilistically samples the current class label distributions and assigns a label to each instance based on its distribution. In step 6, it records these labels, and in step 7 it computes the relational features given the current class labels. In step 8, it re-computes the posterior class label probabilities given these relational features. The process then repeats. When the process terminates, the statistics recorded in step 6 approximate the joint distribution of class labels, which is used in step 9 to identify each instance’s most likely class label. These labels are returned in step 10.

Like ICC, we use  $Z_{\text{treatAbsentAsUnknown}}$  to cautiously handle “absent” links. Gibbs sampling is inherently somewhat cautious as it considers its confidence in estimated test set labels when it re-samples the distribution in step 5, although the classifier still treats all links equally. Unlike with ICC, it’s unclear how to favor the training links while still enabling the re-sampling to properly explore the state space. These properties also apply to other algorithms such as relaxation labeling; we will explore more in future work.

### Computational Complexity Analyses

Both algorithms use space that is linear in the number of instances ( $N$ ). The dominant computation costs for both stem from the relational features computation and instance classification. Typically, instances are connected to a small number of other instances, so the first cost is  $O(N)$ . The classification time per iteration is  $O(N)$  for Naive Bayes, and is  $O(N^2)$  for k-NN. Favoring more confident instances requires sorting the instances by probability with a cost of  $O(N \log N)$ , although classification time usually dominates. Therefore, the overall computational cost per iteration of both algorithms is roughly the same. However, the number of iterations varies significantly across the algorithms. Based on Neville and Jensen (2000), we set  $n=10$  for ICC and found that more iterations did not improve performance. In contrast, Gibbs sampling typically requires *thousands* of iterations. Based on Neville and Jensen (2004), we set  $n=2000$  and ignored the first 200 iterations for “burn-in” (larger  $n$  did not improve performance). The comparatively larger number of Gibbs iterations implies that ICC is much less expensive than PGCC.

### Evaluation

**Hypotheses.** (1) The most cautious version of ICC outperforms its aggressive version, (2) The cautious version of PGCC outperforms its aggressive version.

**Data Sets.** We used the following data sets (see Table 1):

1. **Cora** (McCallum *et al.* 2000): A collection of machine learning papers categorized into seven classes.
2. **CiteSeer** (Lu and Getoor 2003): A collection of research papers drawn from CiteSeer (2006).
3. **WebKB** (Craven *et al.* 1998): A collection of web pages from four computer science departments categorized into six classes (*Faculty*, *Student*, *Staff*, *Course*, *ResearchProject*, or *Other*). *Other* is problematic because it is too general, representing 74% of the pages. Like Taskar *et al.* (2002), we discarded all *Other* pages that did not have at least three outgoing links, yielding a total of 1541 instances of which 30% are *Other*.

In Cora and CiteSeer, links exist between the test and training sets that can help collective inference (see links to “different folds” in Table 1). This was previously described as the “in-sample” classification task. For WebKB, there are no links between the schools, so we measure accuracy on the “out-of-sample” task.

**Table 1.** Data sets summary

Characteristics	Cora	CiteSeer	WebKB
Instances	2708	3312	1541
Avg. links per instance (total)	4.01	2.77	6.59
Avg. links per inst.(to diff. fold)	1.46	0.30	0
Class labels	7	6	6
Non-rel. features available	1433	3703	100
Non-rel. features used	100	100	100
Relational features used	14	12	12
Folds	3	3	4

**Feature Representation.** We focus on collective classification tasks involving data sets that are predominantly textual (e.g., web pages or scientific literature). Our instance representation includes relational and non-relational features, as described below.

*Non-relational (content features):* We use a bag-of-words representation for the textual content of instances. In particular, we use a binary representation where the feature corresponding to a word is assigned *true* if the word occurs in the instance and *false* otherwise. For WebKB, we used all 100 words available in our version of the dataset. For Cora and CiteSeer, we used information gain on the training set to identify and select the 100 highest-scoring words to use as the non-relational features. Using more words did not improve performance.

*Relational features:* We compute relational features like the following:

$$f_B(i) = \text{Neighbors}_B(i) / \text{Neighbors}(i) \quad (1)$$

where  $\text{Neighbors}(i)$  is the number of instances hyperlinked to instance  $i$  that are not labeled *unknown*, and  $\text{Neighbors}_B(i)$  is the number of such instances that currently have the label “B.” If  $\text{Neighbors}(i)$  is zero, then  $f_B(i)$  is set to *unknown* if  $Z_{\text{treatAbsentAsUnknown}}$  is true (see Figure 1) or *false* otherwise. There is one such feature per possible class label. We compute individual features using only incoming and only outgoing links. Thus, for a dataset with six possible class labels, we compute 12 relational features. We also investigated the use of binary features and found the results consistent with those reported below. Hence, we omit further details.

**Classifiers.** We used two classifiers with the algorithms:

1. **NB:** A naïve Bayes classifier (based on the Proximity toolkit developed at the University of Massachusetts Amherst, <http://kdl.cs.umass.edu/proximity>).
2. **k-NN:** A case-based classifier that uses the k-nearest neighbor rule (k-NN). Similarities are computed with a function that treats all features equally, but we experimentally found consistent results with other weighting functions. We used  $k=11$  neighbors for voting.

Gibbs sampling requires that all input probabilities be non-zero, so we smoothed the estimated probabilities.

**Tested Algorithms.** We considered a traditional classification algorithm and the two collective inference algorithms with their cautious variants as follows:

**Table 2.** Average % classification error rate.

Algorithm	Cora		CiteSeer		WebKB	
	kNN	NB	kNN	NB	kNN	NB
NonColl	39.1*	31.8*	39.1*	34.2*	46.0*	44.4
ICC <sub>0</sub>	28.8*	23.3*	31.5*	31.2	43.5*	47.8
ICC <sub>U</sub>	26.9*	21.6*	32.1*	31.1*	39.5	48.5
ICC <sub>U/Tr</sub>	25.1*	21.6*	31.9*	31.2*	39.5	48.5
ICC <sub>U/Tr/C</sub>	<b>21.4</b>	<b>19.7</b>	<b>28.9</b>	<b>30.5</b>	39.4	<b>43.7</b>
Gain <sup>†</sup>	7.4	3.6	2.6	0.7	4.1	4.1
PGCC <sub>0</sub>	26.6*	22.4*	31.7	30.9	44.2*	48.7
PGCC <sub>U</sub>	24.9*	21.2*	31.9	30.9*	<b>39.1</b>	53.2

\* indicates significantly worse behavior than ICC<sub>U/Tr/C</sub>. Higher cross-validation variance makes some comparisons not significant.

<sup>†</sup> indicates gain from caution (ICC<sub>0</sub> - ICC<sub>U/Tr/C</sub>)

1. NonColl: is a traditional (i.e., non-collective) classification algorithm that performs a single run of the base classifier without any relational features.
2. We tested the following 4 variants of ICC ranging from the most aggressive to the most cautious:
  - i. ICC<sub>0</sub>: ICC in its most aggressive form obtained by setting the three cautious parameters to *false*.
  - ii. ICC<sub>U</sub>: A slightly cautious version of ICC obtained by setting  $Z_{\text{treatAbsentAsUnknown}}$  to *true*.
  - iii. ICC<sub>U/Tr</sub>: A version of ICC that is even more cautious than ICC<sub>U</sub>, obtained by setting  $Z_{\text{favorTrainLinks}}$  to *true*.
  - iv. ICC<sub>U/Tr/C</sub>: The most cautious version of ICC; it sets all three cautious parameters to *true* (adding in the favoring of more confident instances).
3. We tested two algorithms that use Gibbs sampling:
  - i. PGCC<sub>0</sub>: PGCC with  $Z_{\text{treatAbsentAsUnknown}}$  set to *false*.
  - ii. PGCC<sub>U</sub>: A more cautious variant of PGCC obtained by setting  $Z_{\text{treatAbsentAsUnknown}}$  to *true*.

**Performance Measure.** We compared all the algorithms for their average classification error rate on the test sets.

**Test Procedure.** We conducted an n-fold cross-validation study for each algorithm and its variants. The Cora and CiteSeer data sets, as provided to us, were split into three roughly equal sized folds that preserved linking within a fold, which we used as is. We did not use randomly generated folds because that could remove the naturally occurring relations, resulting in folds that would be unrealistic for a collective classification task. For WebKB, we conducted a 4-fold cross-validation study, treating each of the four schools as a separate fold.

**Analysis.** We performed independent analyses for each data set and joint analyses by pooling the observations from all the data sets and across both classifiers. Our conclusions are based on one-tailed paired t-tests accepted at the 95% confidence level.

**Results.** Table 2 displays the classification error rates averaged over all the folds for each algorithm. For each data set and classifier, the best result is shown in bold.

*Result 1. The most cautious variant of ICC significantly outperforms its aggressive variant:* Comparing ICC<sub>0</sub> with

ICC<sub>U/Tr/C</sub>, we find that when all 3 cautious behaviors are combined, they reduce classification error by 0.7% to 7.4%, for an average improvement of 3.8%. This improvement is significant in every case except CiteSeer+NB and WebKB+NB. Using a pooled data analysis (not shown in Table 2), ICC<sub>U/Tr/C</sub> significantly outperforms ICC<sub>0</sub> (average errors of 31.7% vs. 35.5% [ $p=0.003$ ]). Therefore, we accept Hypothesis #1.

Examining the individual contribution of parameter settings reveals that the largest error reduction comes from setting  $Z_{\text{favorConfidentInstances}}$  to true. For example, comparing ICC<sub>U/Tr</sub> with ICC<sub>U/Tr/C</sub>, we notice that errors are reduced in all six cases across the three data sets and two classifiers (significantly, for Cora and CiteSeer). A pooled data analysis shows a significant improvement (31.7% vs. 34.1% [ $p=0.025$ ]). The other two parameters also improve performance, but comparatively less in magnitude and less consistently across the data sets. For example, setting  $Z_{\text{treatAbsentAsUnknown}}$  to true (ICC<sub>U</sub>) improves performance in three cases (by 1.7% to 4%), and has negligible impact in the others. The difference is significant for Cora+NB, and for the pooled analysis (34.4% vs. 35.5% [ $p=0.037$ ]). Likewise, the incremental impact of setting  $Z_{\text{favorTrainLinks}}$  to true (ICC<sub>U/Tr</sub>) is small (and not significant) in Cora+kNN and essentially none otherwise. This is because Cora has by far the most links between folds, and WebKB has none (see Table 1).

*Result 2. The cautious variant of PGCC does not outperform the aggressive variant:* Comparing PGCC<sub>0</sub> vs. PGCC<sub>U</sub>, we see that setting  $Z_{\text{treatAbsentAsUnknown}}$  to true has negligible effect for CiteSeer, improves performance for Cora and WebKB+kNN, and hurts performance with WebKB+NB. Overall, we find that treating absent links as unknown was generally helpful, but not in a statistically significant manner. Therefore, we reject Hypothesis #2.

*Result 3. The most cautious variant of ICC significantly outperforms the most cautious variant of PGCC.* ICC<sub>U/Tr/C</sub> almost always outperforms PGCC<sub>U</sub> with the exception of WebKB+kNN. The difference is significant for Cora and CiteSeer+NB. In addition, a pooled analysis shows ICC<sub>U/Tr/C</sub> significantly beats PGCC<sub>U</sub> (31.7% vs. 34.8% [ $p=0.001$ ]). Note that this advantage over Gibbs does not hold for less cautious variants (i.e., ICC<sub>0</sub> and ICC<sub>U</sub>).

*Result 4. Collective classification algorithms, cautious or otherwise, outperform non-collective classification:* We found that, for these classification tasks, collective inference almost always improves accuracy across the two inference algorithms and two classifiers. Using the pooled analysis, the six collective classification approaches shown in Table 2 each significantly outperform NonColl. Considering the data set and classifier combinations individually, WebKB+NB is the only exception. In this combination, the baseline error rate is high, so there is increased potential for decreased performance due to uncertainty in instance labels. Nevertheless, when all three cautious settings are used (with ICC<sub>U/Tr/C</sub>), collective inference slightly reduces the error vs. NonColl.

## Discussion

Overall, cautious behavior for collective inference is highly effective. Within an algorithm family, ICC or PGCC, cautious settings improved performance. In particular, the gains were larger and statistically significant for ICC. Also, the most cautious ICC ( $ICC_{U/TR/C}$ ) outperformed the best PGCC. While further Gibbs tuning (e.g., with random restarts) might marginally improve PGCC's performance, ICC's ability to match or exceed PGCC's results suggests that ICC is a promising alternative.

Our results show that favoring instances with higher confidence class label predictions and favoring links to the training set significantly improves performance. Although Neville and Jensen (2000) previously showed that the former factor could improve accuracy compared to a non-relational classifier (e.g., `NonColl`), to our knowledge, this paper is the first to (1) show that this approach outperforms other collective algorithms (e.g.,  $ICC_U$ ), (2) identify two distinct factors providing its advantage, and (3) evaluate its effects on more than one dataset.

Our findings are inconsistent with those of Lu and Getoor (2003) on the same datasets. They reported no significant improvement from using a confidence-based ordering derived from Neville and Jensen's algorithm. However, our algorithm differs in the way confidence is used. In our approach, early ICC iterations utilize only those instance labels that have high confidence, treating others as *unknown*. In contrast, Lu and Getoor's algorithm instead utilizes all neighboring labels; the only change is in the *order* in which these decisions are made. Our results suggest that explicitly excluding the lower confidence labels, via *unknown* values, is critical for effectively utilizing confidence.

## Conclusion

Collective inference has the potential to substantially improve classification accuracy on interrelated data, but must be carefully applied because uncertain predicted labels can produce incorrect relational features that diminish accuracy. We conjectured that cautious use of such uncertain information could improve performance. On this basis, we generalized two collective inference algorithms to control their degree of caution when using uncertain information. Our results demonstrated that, indeed, "cautious" approaches can be more effective. Furthermore, we showed how the simple ICC algorithm, in its most cautious mode, could outperform Gibbs sampling, a popular yet computationally expensive approach.

Further research is needed to confirm our results using other datasets and collective inference algorithms. Work is also needed to compare the relative performance of different types of cautious behaviors and to explore which types of datasets benefit most from such caution. For instance, how does the amount of label auto-correlation and noise affect performance? Finally, most collective algorithms have an asymmetry: models are trained using fully correct instance labels, yet incorrect labels will be

present during testing. We intend to measure how cautious behaviors naturally compensate for this asymmetry, and also to explore other techniques for ameliorating it.

## Acknowledgements

We thank the Naval Research Laboratory and the U.S. Naval Academy for supporting this work. Thanks to Frederick Crabbe, David Jensen, Donald Patterson, and the anonymous reviewers for their helpful comments. Portions of this analysis used Proximity, an open source software environment from the Univ. of Massachusetts, Amherst.

## References

- Chakrabarti, S., Dom, B., and Indyk, P. (1998). Enhanced hypertext categorization using hyperlinks. *Proceedings of the International Conference on Management of Data* (pp. 307-318). Seattle, WA: ACM.
- CiteSeer (2006). CiteSeer.IST scientific literature digital library. [<http://citeseer.ist.psu.edu>].
- Craven, M., DiPasquo, D., Freitag, D., McCallum, A., Mitchell, T., Nigam, K., and Slattery, S. (1998). Learning to extract symbolic knowledge from the world wide web. *Proceedings of the Fifteenth National Conference on Artificial Intelligence* (pp. 509-516). Madison, WI: AAAI Press.
- Jensen, D., and Neville, J. (2002). Autocorrelation and linkage cause bias in evaluation of relational learners. *Proceedings of the Twelfth International Conference on Inductive Logic Programming* (pp. 101-116). Sydney, Australia: Springer.
- Jensen, D., Neville, J., and Gallagher, B. (2004). Why collective inference improves relational classification. *Proceedings of the Tenth International Conference on Knowledge Discovery and Data Mining* (pp. 593-598). Seattle, WA: ACM.
- Lu, Q., and Getoor, L. (2003). Link-based classification. *Proceedings of the Twentieth International Conference on Machine Learning* (pp. 496-503). Washington, DC: AAAI.
- McCallum, A., Nigam, K., Rennie, J., and Seymore, K. (2000). Automating the construction of internet portals with machine learning. *Information Retrieval*, **3**, 127-163.
- McDowell, L., Gupta, K. M., and Aha, D.W. (2007) Case-based collective classification. In *Proceedings of the Twentieth International FLAIRS Conference*. Key West, FL: AAAI.
- Neville, J., and Jensen, D. (2000). Iterative classification in relational data. In L. Getoor and D. Jensen (Eds.) *Learning Statistical Models from Relational Data: Papers from the AAAI Workshop* (Technical Report WS-00-06). Austin, TX: AAAI.
- Neville, J., and Jensen, D. (2004). Dependency networks for relational data. *Proceedings of the Fourth International Conf. on Data Mining* (pp. 170-177). Brighton, UK: IEEE.
- Neville, J., and Jensen, D. (2005). Leveraging relational autocorrelation with latent group models. *Proceedings of the Fifth International Conference on Data Mining* (pp. 322-329). Houston, TX: IEEE.
- Sen, P., and Getoor, L. (2006). Empirical comparison of approximate inference algorithms for networked data. In A. Fern, L. Getoor, and B. Milch (Eds.) *Open Problems in Statistical Relational Learning: Papers from the ICML Workshop*. Pittsburgh, PA: [www.cs.umd.edu/projects/srl2006](http://www.cs.umd.edu/projects/srl2006).
- Taskar, B., Abbeel, P., and Koller, D. (2002). Discriminative probabilistic models for relational data. *Proceedings of the Eighteenth Conference on Uncertainty in Artificial Intelligence* (pp. 485-492). Edmonton, Canada: Morgan Kaufmann.