Social Network Classification Incorporating Link Type Values

Raymond Heatherly
Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Email: rdh061000@utdallas.edu

Murat Kantarcioglu
Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Email:muratk@utdallas.edu

Bhavani Thuraisingham
Jonsson School of Engineering and Computer Science
The University of Texas at Dallas
Email:bxk043000@utdallas.edu

***Abstract*—Classification of nodes in a social network and its applications to security informatics have been extensively studied in the past. However, previous work generally does not consider the types of links (e.g., whether a person is friend or a close friend) that connect social networks members for classification purposes. Here, we propose modified Naive Bayes Classification schemes to make use of the link type information in classification tasks. Basically, we suggest two new Bayesian classification methods that extend a traditional relational Naive Bayes Classifier, namely, the Link Type relational Bayes Classifier and the Weighted Link Type Bayes Classifier. We then show the efficacy of our proposed techniques by conducting experiments on data obtained from the Internet Movie Database.**

I. Introduction

Social Networks are platforms that allow people to publish details about themselves and to connect to other members of the network through links. Recently, the population of such online social networks has increased significantly. For instance, Facebook now has over 150 million users[1]. Facebook is only one example of a social network that is for general connectivity. Other instances of social networks are LinkedIn (professional), Last.fm (music), orkut (social), aNobii (books), and the list continues.

Each of these networks allow users to list details about themselves, and also allows them to not specify details about themselves. However, these hidden details can be important to the administrators of a social network. Most of these sites are free to the end user and are thus advertising-supported. If we assume that advertisers want to reach the people most likely to be interested in their products, then identifying those individuals becomes a priority for maintaining much-needed advertisers. However, by just using specifically defined information, the site may be missing a large number of potentially interested users. Taking the specified knowledge from some users and using it to infer unspecified data may allow the site to extend its target audience for particular advertisements.

Of course, the implications of classification in social network data extend far beyond the simple case of targeted advertising. For instance, such ideas could be used for addressing classification problems in terrorist networks. By using the link structure and link types among nodes in a social network with known terrorist nodes, we can attempt to classify unknown nodes as terrorist or non-terrorist. In such a classification process, the type of the link shared among nodes could be really critical in determining the final success of the classifier. For example, assume that there exist two different individuals, Alice and Bob, that are linked to some known terrorist, Malory. Furthermore, assume that Alice works at the same place as Malory, but they are not friends. (i.e., Alice and Malory have a relationship. The type of relationship is "colleague"). In addition assume that Malory talks frequently on the phone with Bob and they are friends (i.e., Bob is linked to Malory and the link type is "friend"). Given such a social network, to our knowledge, all the existing classification methods just use the fact that individuals are linked and they do not use the information hidden in the link types. Clearly, in some cases, one link type (e.g., friendship) can be more important than other link types for increasing the accuracy of the classification process. Therefore, social network classification techniques that consider link types could be very useful for many classification tasks.

To address the above problem, we choose the network-only Naive Bayes classifier as our starting point, since it is shown in [1] that relational Naive Bayes classification combined with collective inference techniques provide an efficient solution with acceptable accuracy in practice. We modify this relational Naive Bayes classifier to incorporate the "type" of the link between nodes into the probability calculations. First, we devise a naive Bayes classification method in which all link types are equally important and we then refine this method by adding an additional granularity to the calculation method. To our knowledge, this is the first paper in using link types for relational Naive Bayes classification.

Our paper is organized as follows. In section II, we provide a brief overview of the related work. In section III, we discuss our relational Naive Bayes methods that incorporate link types. In section IV, we show the effectiveness of our learning method on real world data set. Finally, we conclude with the brief discussion of the future work.

II. Related Work

In [2], the authors use iterative classification techniques to improve accuracy of classification methods of social networks.

[1]http://www.facebook.com/press/info.php?statistics

This approach is mirrored in [1], and the authors extend the work with a full comparison of several relational classification methods and several collective inference techniques. However, while [1] begins to acknowledge the use of link types, in their experiments using the IMDb dataset, they generate separate relational graphs based on this link type. Unfortunately, they consider only one link type in each experiment. Similarly, [3] considers the problem of information revelation on the Facebook social network, giving a focus to the users' ability to decrease the amount of information they reveal, but highlights the relatively small number of users who actually do so.

In [4], the authors use hypothetical attributes from the real-world dataset of LiveJournal to construct a Bayesian Network which they use to analyze their algorithm. However, they do not consider any aspect of link types. [5] discusses methods of link re-identification. However, in our data, the links are not explicitly defined and are instead inferred directly from the stated facts. Anonymization of the data that allows us to generate the relationships would make the data meaningless.

In [6], Jensen et al analyze the importance of collective inference methods on relational networks. Senator gives a general review of the past research in link mining and link analysis, and goes on to issue a call for a unified framework to support a variety of data representations and for the analysis tools in [7]. Getoor and Diehl, in [8] offer a survey of the various tasks performed on both hetero- and homogeneous social networks.

In [9], Sweeney describes several problems and poses a potential solution to naively generating social network from data that did not specifically come from a social network. The author indicates that the use of a form of privacy-enhanced linking in algorithm development can be used to guarantee individual privacy while advancing the cause of technological research.

In [10], Chau and Faloutsos implement a method of detecting fraudulent transactions in the online auction site E-Bay. By modeling transaction histories of buyers and sellers, they are able to use characteristic features and design a decision tree that is able to detect fraudulent activities with an 82% accuracy.

In [11] and [12], the authors use relational Bayesian classification techniques in an attempt to determine private information from a social network. Finally, in [13], authors conduct experiments and build a system to classify both relational and attribute-based models. They then show the accuracy of their system on a set of constructed data.

III. Learning Methods

Currently, classification in social networks is done in a variety of ways. The simplest of these is to use the data about nodes for which the label is known, and to use a simple classification mechanism, such as a naive Bayes Classifier, to classify based on only those attributes that are local to the node being classified. Methods that classify in this manner are considered *local classifiers*.

Another method is referred to as *relational classifiers*. These classification algorithms model the social network data as a graph, where the set of nodes are a specific, homogeneous entity in the representative network. Edges are added based on specific constraints from the original data set. For instance, in Facebook data, the edges are added based on the existence of a friendship link between the nodes. In [1], the authors determine a link between two nodes in their IMDb dataset if the two nodes share the same production company. Once this graph structure is created, a classification algorithm is then applied that uses the labels of a node's neighbors to probabilistically apply a label to that node. These algorithms use the theory of *homophily*, that is, that a node is more likely to associate with other nodes of the same type, to perform their calculations.

However, one of the problems with even relational classifiers is that if the labels of a large portion of the network are unknown, then there may be nodes for which we cannot determine a classification. To truly represent homophily, the classified nodes should be used to re-assess the classifications of its neighbors. This is the thought behind *collective inference*. These algorithms use a local classifier to create a set of class priors for every node, to ensure that each node has an initial "guess" of a classification. Then, the algorithm uses a relational classifier to generate a probability assignment for each node based on those class priors. This allows us to use a relational classifier and ensure that each node will have a classification. The collective inference algorithm specifies a weighting scheme for each iteration. It also specifies either a number of iterations to run for or to wait for convergence in the classification.

Because we focus our efforts on improving the overall relational classification by increasing the data available for this class of classifiers to use, we use a single local classifier and vary the relational classifiers. Each of these methods is based on the traditional naive Bayes classifier.

A. Local Classification

As shown in [11], using a simple Bayes classifier is an effective method of discerning private information from a social network. Because of this finding and its relatively low computational requirements, we use a basic Naive Bayes classifier. We use the classifier to determine the probability that a particular node, x_i, is of a particular class, c_i, given the entire set of its traits, $\mathcal{T}$ by the formula

$$\begin{aligned} &Pr(x_i = c_i|\mathcal{T}) = \\ &\frac{Pr(\mathcal{T}|x_i = c_i)Pr(x_i = c_i)}{Pr(\mathcal{T})} = \\ &Pr(x_i = c_i) \times \prod_{t_i \in \mathcal{T}} \frac{Pr(t_i|x_i = c_i)}{Pr(t_i)} \end{aligned} \tag{1}$$

B. Network-only Bayes Classification

This is the first, and most basic, of the three relational classifiers that we examine here. The general nBC assumes that all link types are the same, and that the probability of

a particular node's class is influenced by the class of its neighbors. We use the local classifier to assign an inferred prior of each node in the test set. Since the details of a particular node are factored in when we establish these priors, we do not duplicate this in the nBC calculations. We alter the probability calculations as follows:

$$Pr(x_i = c_i|\mathcal{N}) = \frac{Pr(\mathcal{N}|x_i = c_i)Pr(x_i = c_i)}{Pr(\mathcal{N})} = Pr(x_i = c_i) \times \prod_{n_i \in \mathcal{N}} \frac{Pr(n_i|x_i = c_i)}{Pr(n_i)} \quad (2)$$

This method is a basic classifier, yet Chakrabarti et al and Macskassy and Provost ([14], [1], respectively), both use nBC to effectively classify nodes on a variety of data sets. Because of their findings, we use this algorithm as the basis for ours.

C. Link Type nBC

The next relational classifier that we test is the first to include link types. We noticed that there were no classification algorithms to specify constraints about what type of link two individuals shared, in an important way, for the probability calculations. We believed that including these differences when determining probabilities could be an important and useful extension. For instance, consider Ron Howard. As a writer, director, producer, and actor, he is linked to over a hundred different television shows and movies. However, when classifying whether a movie would be a success, there must be a difference in how important his role in a production is to calculate his weight on the movie's success.

We represent this by including the link type as an additional parameter to a Naive Bayes classification. Whereas originally, we define the probability of any specific node, x_i to be in a particular class, c_i, to be $Pr(x_i = c_i|\mathcal{N})$ – that is, the probability of a node being in a class is dependent only upon its neighbors – we now define the probability to be $Pr(x_i = c_i|\mathcal{N}, \mathcal{L})$. That is, the probability of any particular node being in a class is determined by its neighbors, and the set of links that define those neighbors. So, we amend the traditional Naive Bayes calculation to be:

$$Pr(x_i = c_i|\mathcal{N}, \mathcal{L}) = \frac{Pr(\mathcal{N}, \mathcal{L}|x_i = c_i)Pr(x_i = c_i)}{Pr(\mathcal{N}, \mathcal{L})} = \prod_{\substack{n_i \in \mathcal{N} \\ l_i \in \mathcal{L}}} \frac{Pr(n_i, l_i|x_i = c_i)Pr(x_i = c_i)}{Pr(n_i, l_i)} \quad (3)$$

D. Weighted Link Type rBC

The final relational classifier considers that certain link types are indicative of a stronger relationship. For instance, our IMDb data set contains data about all of the crew of a show. This includes the directors, producers, actors, costume designers, grippers, and special effects technicians. We theorized that not all of these jobs can be equally important. So, we alter our Link Type rBC to include weights. To this, we add the idea of variable weights to the calculation in Equation 3.

$$Pr(x_i = c_i|\mathcal{N}, \mathcal{L}) = \prod_{\substack{n_i \in \mathcal{N} \\ l_i \in \mathcal{L}}} [\frac{w_i}{W} \times \frac{Pr(n_i, l_i|x_i = c_i)Pr(x_i = c_i)}{Pr(n_i, l_i)}] \quad (4)$$

where w_i is the weight associated with link type l_i, and W is the sum of all weights for the network.

E. Relaxation Labeling

We choose to use Relaxation Labeling as described in [1], a method which retains the uncertainty of our classified labels. Relaxation Labeling is an iterative process, where at each step $i+1$ the algorithm uses the probability estimates, not a single classified label, from step i to calculate probability estimates. Further, to account for the possibility that there may not be a convergence, there is a decay rate, called α set to 0.99 that discounts the weight of each subsequent iteration compared to the previous iterations.

We chose to use Relaxation Labeling because in the experiments conducted by Macskassy and Provost[1], Relaxation Labeling tended to be the best of the three collective inference methods.

We decided to choose the best of the three methods as reported in [1] and use that to focus on our new relational classifiers.

IV. Experiments

The data used in our experiments was gathered from the Internet Movie Database (IMDb). This decision was made based on the importance of not only having links between members in our social network representation, but on the ability to specifically define what specific relationship is indicated by them. Although, we considered using data from the Facebook online social network, we discovered that while Facebook allows users the ability to specify the relationship type linking friends, most relationships are not defined. Rather than add additional inference operations that could cast doubt on the validity of the relationship types, we chose to use a dataset where relationship types are given.

We implemented a crawler using Java 1.6 that crawled the site and parsed information from a selection of pages of all movies. The pages we selected were Full Cast and Crew, Main Details, Company Credits, Business/Box Office, and Awards. We store the data about movies in an RDF datastore which gives us a table of 3-tuples, where each tuple is a single fact about a movie. For instance, the movie *Transformers*, if assigned the IMDb unique ID of tt0418279, would be represented by the tuple <tt0418279, Title, Transformers>. It is important to note that when we record facts about individuals who participate in a movie, we use their IMDb unique ID, so there is no confusion between people who may have the same name.

We then define two movies to be related if they share at least one individual in the same job. For instance, *Batman*

Begins would be related to *The Dark Knight* because they share (as a single instance) Christopher Nolan in the job *Director*. Similarly, *Batman Begins* and *Equilibrium* would be related because they both have Christian Bale as an actor in them, even though he does not play the same role in both films. Additionally, *Batman*, *Batman Begins*, and *Batman and Robin* are all related because they share the same character – Batman. All fields were initially used to create relationships. however, we discovered in experimentation that the use of the two languages and properties <*, Language, English>and <*, Country, USA>resulted in a very tight clique where a majority of movies were all related to each other. As a result of this, we removed those two entries for all movies to reduce the number of relationships in our dataset.

A. Datastore alterations

We began with all information stored in a single table in RDF-format. In the process of experimentation, we made several alterations to this in order to increase the efficiency of our classification routine.

The first change we made was to create a separate table to store all of the data about those nodes for which we have the Earnings property recorded. The motivation for this came from the queries used to generate relationships and the local classifier. To ensure that only relationships among these "valid" nodes were considered, we either maintained a full list of valid nodes in memory to check against, or used queries with a series of joins. However, both of these steps required that we conduct them each time a relationship was found in every iteration. We chose to implement a pre-processing step that pulls the classification criteria from a configuration file, and inserts all the tuples of nodes that meet the requirements into the validnodes table. We then use this as our main datastore for the remainder of experiments. Also, the preprocessing step adds a tuple defining the field that the validnodes table is created on. This allows our implementation to intelligently decide if this pre-processing step needs to be conducted or if it may be skipped in later iterations. As a direct benefit of this step, our local classification step requires considerably less time to run.

The last alteration that we made was an additional pre-processing step to pre-compute the relationships in our dataset. We decided to pre-compute the relationships because of the time involved in determining them. Each individual test in each experiment spent time re-discovering the same static relationships. To reduce this time, we added a pre-computing step that creates a table to maintain a list of all relationships between nodes in our *validnodes* table. We maintain this as another 3-tuple of the form <NodeA, NodeB, <Relationships>>. Each row indicates that there exists at least one type of relationship between NodeA and NodeB. The link types and the values of each link type are stored in a vector. So, for instance, let us use the earlier examples of *Batman Begins* and *The Dark Knight*. We would have an entry of the form <"Batman Begins", "The Dark Knight", <Director=Christopher Nolan; Actor=Christian Bale>>. This allows us to generate one relationship table that we can then use various ways in all of our experiments.

B. Experimental Setup

As our classification criterion, we follow the example in [1] and use earnings figures in an attempt to determine whether a movie will make in excess of $2 million. We begin with a set of 259,937. We then eliminate those movies which do not have a figure recorded for Earnings. This leaves us with a set of size 5,382. Of these nodes, 3,324 earned over $2 million. This gives us a baseline accuracy of 59.9% for simply guessing that each movie will make over $2 million.

We conduct experiments at each ratio of 10/90, 20/80, . . ., 90/10 for training data versus test data. To do this, we randomize the order of the nodes and then construct the partition at the appropriate spot for the ratio we are testing at the time. Once we have these two random sets, we conduct four experiments using the Relaxation Labeling method of Collective Inference. For each test set, we consistently use a Naive Bayes classifier as the local classifier, but we vary the relational classifier. We repeat each test 20 times and take an average of all runs for the presented results.

C. Results

Our first experiment was to use only the local Bayes Classifier with Relaxation Labeling to establish a baseline accuracy of a non-relational classifier on our particular dataset. As can be seen in Figure 1, initially, increasing the ratio of labeled nodes to unlabeled nodes drastically increases the classification accuracy. However, after 40% of the nodes are labeled, the gains from additional nodes in the training set are minimal. This does show that even though we do not consider any relationships at all, by simply using a method of supervised learning, we can improve on the naive method of guessing the most populous group. This improvement is evident even in a situation where most of the class values for the nodes are unlabeled.

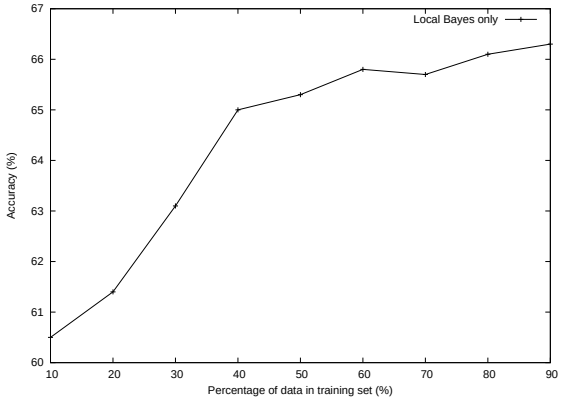

Fig. 1. Local Bayes only

Our second experiment was conducted to establish a performance baseline of an existing relational classifier on our extended dataset. We compare these results to [1]. Specifically,

the *imdb_prodco - nBC* results. While our classifier does under perform in comparison to theirs, as shown in Figure 2, we do still see improvements upon the local Bayes classifier. We believe that the performance degradation in this result is because of overfitting. Consider that their experiment was conducted using only one attribute – the production company – as the determinant of relationships between movies, whereas we consider all attributes to be indicative of relationships. This large number of relationships appears to inject a higher degree of error into our trials as opposed to simply using a single attribute.

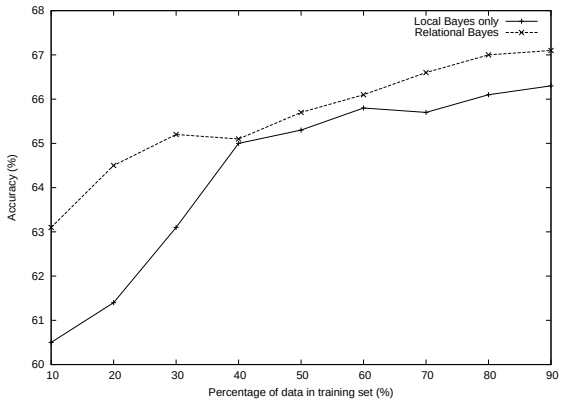

Fig. 2. Relaxation Labeling using Relational Bayes Classifier

In our third experiment, we use the values for the link types. In this set of tests, we used the probabilities shown in Equation 3 to consider the link type as an additional parameter. We see quite clearly in Figure 3 that including the link types in the calculations increases our accuracy dramatically. Even when we had the least number of nodes in the training set, we achieved over a ten percentage point increase in accuracy over the generic relational Bayes Classifier. We believe that because we give some difference to the link types, this allows our classification method to make up for the performance loss in the previous experiments.

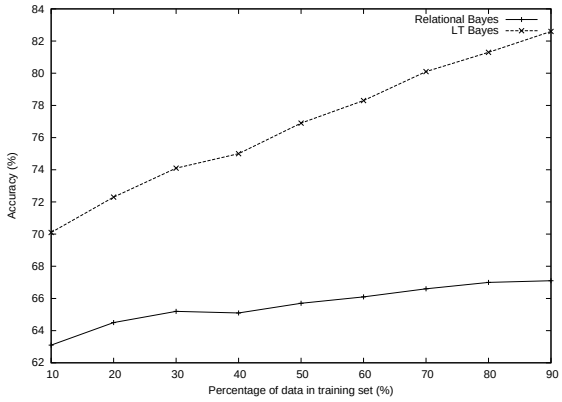

Fig. 3. Relaxation Labeling using Link Type Relational Bayes Classifier

In our final series of experiments, we use the weighted Link Type Relational Bayes Classifier. In [15], the authors identify seven components of a movie that they use to design a Decision Support System for a Hollywood production company. The areas they identify are: MPAA Rating, Competition, Star Value, Genre, Special Effects, Sequel, and Number of Screens. Some of these attributes, such as Competition, Special Effects, and Number of Screens, are opinion-based values, and are difficult to quantify for a dataset spanning the number of years ours covers. We took the their description of Star Power, and considered what current trailers for movies were listing as being an inducement for the movie. We divide the jobs in to three groups. The first group is composed writers, directors, and actors. The second group is made up of technical crew. That is, editors, costume, wardrobe, camera, and electrical. The third group is comprised of the remaining roles: producers, production company, special effects studio, etc. To test our division of roles, we conducted individual tests on a 50/50 division into training and test sets. That is, half of the nodes in the data set for these tests was unlabeled. For these tests, we specify a weight of x for the actors, directors, and writers; and a weight of y for the technical crew. The results of this experiment are shown in Table I

	y = 0.5	y = 2
x = 0.5	74.1	72.0
x = 2	83.2	78.6

TABLE I
SUMMARY OF ACCURACY TESTS WITH VARIED WEIGHTS

As shown in the table, our assumptions about the importance of respective jobs was confirmed. In those experiments where we increased the weight value of actors, directors, and writers, the accuracy of our classification calculations increased. Similarly, when we increase the weight value of the technical crew, the accuracy decreases from 74.1% to 72% when $x = 0.5$ and from 83.2% to 78.6% when $x = 1$.

With this consideration, we give all writers, actors, and directors a weight of 2, and specify that technical crew is given a weight of 0.5. We have a default weight of 1.0 for all other jobs, such as animation, music, choreography, etc. We show in Figure 4 that using these weights, we achieve substantial performance gains. Even at a relatively low percentage of known data, we are able to accurately classify nodes over 80% of the time, and at 90% of nodes labeled, our accuracy is slightly over 85%. These results indicate an improvement over the results shown in [1], where the accuracy does not seem to reach above 80% for any of the collective inference methods when using the IMDb dataset.

We believe that these figures make it apparent that with even a small amount of domain knowledge, we can use that knowledge to devise weights for link types to increase the accuracy of classification algorithms .

V. CONCLUSION AND FUTURE WORK

We present for ease of comparison a summarization of all of our experiments in Figure 5. We believe that it shows that our enhancements to the traditional Relational Bayes Classifier,

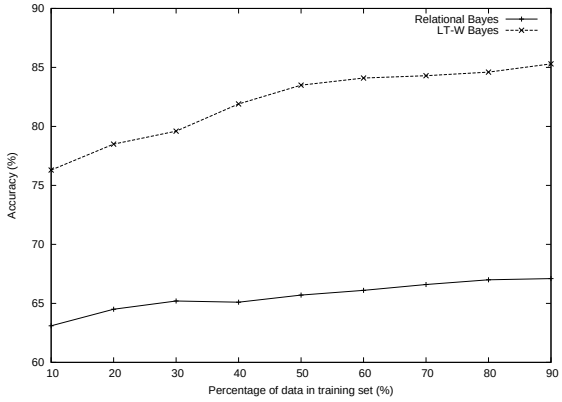

Fig. 4. Relaxation Labeling using Weighted Link Type Relational Bayes Classifier

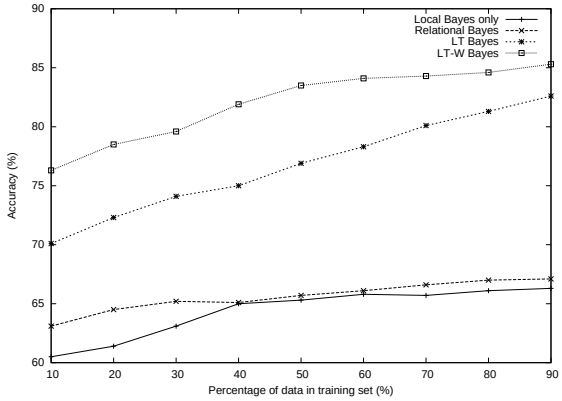

Fig. 5. Comparison of all experiments

which we call Link Type rBC and Weighted Link Type rBC, are both successful extensions to the well-researched area of classification in social networks. We have shown how our extensions have improved upon the implementation of the traditional relational Bayes Classifier on our dataset, and how the overall accuracy compares to similar experiments done on a similar dataset in other research work. We believe this is a solid foundation for further examination of the inclusion of link types in social network classification research.

In the future, we believe that we need to apply these new calculations to other domains in order to determine when these new methods may be applied or when we must use the more traditional rBC implementation. Also, additional research should be conducted in programmatically determining weights for the wltrBC, perhaps by using social network metrics.

VI. Acknowledgments

This work was partially supported by National Science Foundation Grants Career-0845803, CNS-0716424 and Air Force Office of Scientific Research MURI Grant FA9550-08-1-0265. The authors would also like to thank Dr. Art Becker for useful discussions.

References

[1] S. A. Macskassy and F. Provost, "Classification in networked data: A toolkit and a univariate case study," *J. Mach. Learn. Res.*, vol. 8, pp. 935–983, 2007.

[2] J. Neville and D. Jensen, "Iterative classification in relational data," 2000. [Online]. Available: http://citeseer.ist.psu.edu/neville00iterative.html

[3] R. Gross, A. Acquisti, and J. H. Heinz, "Information revelation and privacy in online social networks," in *WPES '05: Proceedings of the 2005 ACM workshop on Privacy in the electronic society*. New York, NY, USA: ACM Press, 2005, pp. 71–80. [Online]. Available: http://dx.doi.org/10.1145/1102199.1102214

[4] W. Xu, X. Zhou, and L. Li, "Inferring privacy information via social relations," in *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, 2008, pp. 525–530. [Online]. Available: http://dx.doi.org/10.1109/ICDEW.2008.4498373

[5] E. Zheleva and L. Getoor, "Preserving the privacy of sensitive relationships in graph data," 2008, pp. 153–171. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-78478-4_9

[6] D. Jensen, J. Neville, and B. Gallagher, "Why collective inference improves relational classification," in *In Proceedings of the 10th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004, pp. 593–598.

[7] T. E. Senator, "Link mining applications: progress and challenges," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 76–83, 2005.

[8] L. Getoor and C. P. Diehl, "Link mining: a survey," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 3–12, 2005.

[9] L. Sweeney, "Privacy-enhanced linking," *SIGKDD Explor. Newsl.*, vol. 7, no. 2, pp. 72–75, 2005.

[10] D. H. Chau and C. Faloutsos, "Fraud detection in electronic auction," in *In European Web Mining Forum at ECML/PKDD*, 2005.

[11] J. He, W. Chu, and V. Liu, "Inferring privacy information from social networks," in *Proceedings of Intelligence and Security Informatics*, Mehrotra, Ed., vol. LNCS 3975, 2006.

[12] W. Xu, X. Zhou, and L. Li, "Inferring privacy information via social relations," in *Data Engineering Workshop, 2008. ICDEW 2008. IEEE 24th International Conference on*, 2008, pp. 525–530.

[13] P. A. Flach and N. Lachiche, "Naive bayesian classification of structured data," *Mach. Learn.*, vol. 57, no. 3, pp. 233–269, 2004.

[14] S. Chakrabarti, B. Dom, and P. Indyk, "Enhanced hypertext categorization using hyperlinks," in *SIGMOD '98: Proceedings of the 1998 ACM SIGMOD international conference on Management of data*. New York, NY, USA: ACM, 1998, pp. 307–318.

[15] D. Delen, R. Sharda, and P. Kumar, "Movie forecast guru: A web-based dss for hollywood managers," *Decis. Support Syst.*, vol. 43, no. 4, pp. 1151–1170, 2007.