J. Neville and D. Jensen (2002). Supporting relational knowledge discovery: Lessons in architecture and algorithm design.
Papers of the ICML 2002 Workshop on Data Mining Lessons Learned.

Supporting Relational Knowledge Discovery: Lessons in Architecture and Algorithm Design

Jennifer Neville JNEVILLE@CS.UMASS.EDU
David Jensen JENSEN@CS.UMASS.EDU
Knowledge Discovery Laboratory, Department of Computer Science, University of Massachusetts, 140 Governors Drive, Amherst, MA 01003 USA

Abstract

This paper discusses a few of the lessons we have learned developing a relational knowledge discovery system. The relationships among data instances in relational data provide extra information for "mining." This additional information has the potential to greatly improve the quality of learned models. However, the dependencies among instances in the data also introduce new statistical challenges for learning algorithms. Relational data provide an ideal environment in which to examine a central challenge of knowledge discovery – its "chicken and egg" character. Data representation can impair the ability to learn important knowledge, but knowing the "right" data representation often requires just that knowledge. With relational data, representation is often a choice; many alternate views of the data provide abundant fodder for reasoning about transformations. In light of this, we discuss representation and design choices that support a co-evolutionary process of knowledge discovery and data transformation in relation data.

1. Introduction

Relational knowledge discovery is a new frontier of data mining that is just starting to be explored. Relational data offer a wealth of previously untapped information to exploit during the discovery process. However, along with new opportunities for discovery, the distinctive characteristics of relational data also present several unique challenges. Relationships in the data represent interdependencies among instances and these dependencies can make it difficult to learn accurate probabilistic models of the data. Also, there are often many alternative ways to represent any given set of relational data. Knowing the "right" representation can be crucial to the discovery process but often this knowledge is not known a priori and needs to be learned during analysis. In order to exploit the opportunities offered by relational data, these issues should inform the design of both the architecture and the algorithms of relational knowledge discovery systems.

The lessons we report have been amassed throughout the process of developing and applying PROXIMITY, a system for relational knowledge discovery[1]. PROXIMITY is a set of tools that operate on a graph database designed to support the process of knowledge discovery in relational data. The tools of PROXIMITY provide methods for an iterative process of attribute creation, model learning and inference. One of the goals of the system is to support a transformative approach to knowledge discovery, which will be discussed in a later section.

The majority of data routinely captured by businesses and organizations are relational, yet over the past decade most data mining research has focused on propositional data. Relational data offer unique opportunities to boost the accuracy of learned models and improve the quality of decision-making if the algorithms can learn effectively from the additional information the relationships provide. Recent efforts to modify traditional data mining techniques for relational data include modified Bayes classifiers, decision trees and association rules (Dzeroski & Lavrac 2001). These algorithms have been successfully applied across a range of applications but the explosive growth of structured data suggests more work is needed in this direction.

Over the last two decades much of the work in the machine learning and knowledge discovery communities has focused on non-relational data, where instances are assumed to be identical and independently distributed (i.i.d.). Relational data violate this assumption. Relationships among instances often reflect dependence among instances, and the instances are often heterogeneous instead of homogeneous. The assumption of independence is one of the most deeply buried assumptions in machine learning techniques and we need to fully understand the effects of such dependencies not only on relational model-building processes, but also on evaluation of these techniques.

To a large extent, current research in relational knowledge discovery is focused on learning from the data and the structure of the relations. This is an important endeavor

[1] For additional details on PROXIMITY, see <http://kdl.cs.umass.edu>.

and it deserves our attention, but there is another aspect to the discovery process for which there is little support in current technologies – data transformation. Data selection and transformation are essential to the successful application of knowledge discovery algorithms. The standardization of propositional data has limited our view of data transformation in the past, but as we move to relational data representations, this issue should return as a consideration.

1.1 Lessons Learned

The aim of this paper is to focus attention on the lessons that we have learned from analyzing a number of relational data sets. These lessons include both key new technical ideas, some of which are yet to be fully explored, and system design choices that we have found to be helpful in our analyses. First, we discuss representation and argue for the utility of a simple graph representation. Second, we characterize the space of relational features and outline the vast number of potential features that are available to relational learners. Third, we define relational autocorrelation, a characteristic of relational data that we have found to be ubiquitous in relational data. Fourth, we argue that relational data mining systems should be designed to support transformative learning in order to coevolve knowledge and representation. Finally, we close with a short discussion of statistical issues resulting from the lack of independence in relational data that should influence algorithm design, and we point to papers treating these issues in greater detail.

The lessons contained in this paper are relevant to researchers working with both relational and propositional data. Many of the data sets considered to be i.i.d. are in fact flattened relational datasets. For example, medical records are often used to build models to predict disease given the unique symptoms of the patient. These records might include attributes that indicate the blood type of the patient's parents as well as the family history of the particular disease. These attributes are examples of information that may be better represented explicitly as relations. It is also possible that many of the patients live in the same area or work in the same building, and that this in fact determines their illness shared cause. In this case, the records are not independent even though they are represented propositionally. Researchers working with propositional data should be aware of problems that may be present if statistical dependencies inadvertently exist in their datasets. The design principles and representation issues discussed in this paper are also intended for a general audience. The intimate connection between the process of discovery and data transformation should be reflected not only in approaches to data collection but also in methods for data storage and access.

Although we will use the 2001 KDD Cup gene data (described in the next section) to provide specific examples throughout this paper, the lessons have been garnered through work in many different relational domains, including fraud detection, citation analysis, financial and corporate data, as well as movie data.

2. 2001 KDD Cup

The examples we provide are drawn for our experience participating in the 2001 KDD Cup (Cheng et al. 2002). For the past 6 years, the KDD Cup has been held in conjunction with the ACM SIGKDD International Conference on Knowledge Discovery & Data Mining. One specific goal of the competition is to provide the KDD community with a common laboratory where current research can be evaluated on practical problems. Last year was the first time that a relational data set was included in a challenge problem, indicating an increase in both the awareness of, and the need for, focused research efforts in relational discovery systems.

Rapid advances in genome mapping have increased the interest in mining data from biological domains; consequently, the data for the competition were drawn from this domain and in particular, the relational tasks were taken from the field of functional genomics. The competition was composed of three classification tasks on biological datasets; two of these tasks involved a dataset containing information about the yeast genome at both the gene level and at the protein level. Genes code for proteins and these proteins localize in various parts of the cell and interact with each other to perform various functions. For simplicity, the rest of this paper will refer to 'genes' only and treat gene as synonymous with protein even though information in the data refer to both proteins and genes.

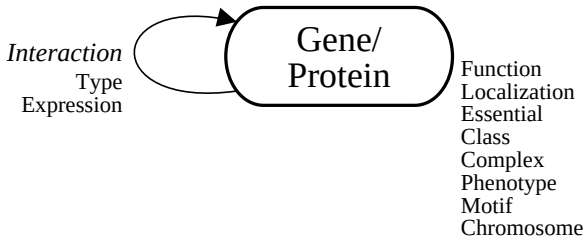

Figure 1: Gene data schema.

Figure 1 shows the data schema for the gene data from the 2001 KDD Cup competition, as represented in PROXIMITY. The data contain information about 1243 genes from the yeast genome with 1734 symmetric interaction links. The average number of interactions per gene is 2.6 (min=0,max=20). The training set consisted of 862 genes; class label information was withheld for 381 randomly selected genes in the test set. The tasks were to predict function and localization for each of the genes in the test set. There are 15 functional categories and each gene can be associated with more than one function. The average number of functions per gene is 2.6

(min=1,max=6). There are 15 locations and each gene localizes in a single location. There are six additional attributes intrinsic to genes and two attributes concerning the interactions between genes. These data illustrate the need for representing attribute information on links. Type is a discrete attribute characterizing the gene interactions. Expression is a continuous attribute in the range [-1,1] measuring the strength of the interaction between the genes.

Tasks 2 & 3 of the competition were to predict function and localization of these proteins; we placed 12 out of 41 for function prediction and 10 out of 45 for localization prediction. We used PROXIMITY to construct 124 relational features in the gene data, and then built Simple Bayesian classifiers with these features. Our test set accuracy for predicting function was 92.6% (1st place: 93.6%), for localization it was 66.14% (1st place: 72%).

3. Data Representation

3.1 Relational Representation

A common approach to learning from relational data is to "propositionalize" the data rather than retain its inherent structure. A great deal of research has been conducted on machine learning techniques for propositional data, and it is often possible to build accurate models of relational data by flattening the data and applying these algorithms. Flattening refers to the process of making each instance identical by either duplicating or aggregating the relational information. For example, the gene data could be flattened into identical instances by taking the average/modal attribute values of a gene's linked genes and appending these new aggregated attributes to the list of intrinsic attribute for each gene. However, flattening relational data removes the richer relational structure and in doing so, may impair learning. In addition, flattening has many other associated problems, which could result in incorrect statistical inferences and/or impaired learning.

Specifically, flattening relational data can result in a combinatorial explosion in either the number of instances or the number of attributes, depending upon whether one decides to duplicate or aggregate. However, a more serious issue lies in the fact that both duplicating and aggregating have the potential to produce biased parameter estimates (Jensen 1998). Flattening destroys any record of the relationships among instances; because the flattened data cannot account for dependencies in the original dataset, estimates of statistical significance on flattened data may be severely biased. Removing the relationships in the data also isolates instances from information about the predictions made on other objects. This inferential isolation eliminates the possibility of using predictions about related instances to inform inferences about other instances. Figure 2 shows that genes cluster by both location and function. Flattening the gene data would preclude the application of an iterative classification technique (Neville and Jensen 2000) that uses predictions made with high confidence to improve predictions in later iterations or other approaches that use relational structure to improve inference (e..g, Friedman, Getoor, Koller, and Pfeffer 1999). Flattening may also result in a representational mismatch of knowledge. Many simple relational concepts are extremely complex to represent in propositional form, yet effective learning requires that exactly the right attributes be identified up front.

One of the main drawbacks of a flattened representation is loss of information. The lost information cannot be reconstructed if it is later determined useful or necessary for the discovery process. However, propositionalization of relational data for learning is often a good approach used by many in the community, including ourselves. In fact, both winning teams in the 2001 KDD Cup used this approach. Maintaining a relational representation of the data does not prohibit flattening dynamically when necessary to learn models and allows for greater flexibility as research progresses in relational model building techniques.

3.2 Simple Graph Representation

We advocate using a simple graph structure to represent relational data. Several nearly equivalent formalisms exist for representing relational data sets including graphs, database tables and first-order logic statements. However, graph representations facilitate reasoning about networks of objects and support the flexible schemas described in the next section.

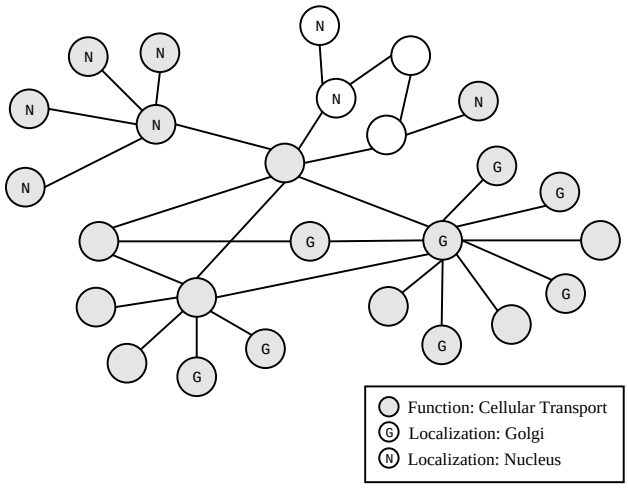

Figure 2: Gene data fragment.

PROXIMITY represents data using objects, links and attributes. Objects are used primarily to represent people, places, and things. Links represent relationships between objects, such as 'mother-of' or 'made-in' or 'part-of.' Attributes represent basic characteristics of items, either objects or links. Attributes have a name (e.g. 'City') and values (e.g. 'Amherst'). Attributes may be associated with

objects and/or links and it is possible for many attribute values to be associated with any one item. In addition to these basic data structures, PROXIMITY can also represent views of the graph as collections of subgraphs. Figure 2 contains a fragment of the yeast gene data from the 2001 KDD Cup as an example. Nodes in the graphs represent genes, and links represent interactions between the proteins that the genes produce.

3.3 Flexible Schema

The PROXIMITY data schema is much more flexible than those commonly used in relational databases. A traditional database schema assumes homogenous object types, storing records for each type of object in a separate table with a fixed numbers of attributes. Once the schema is specified it can be difficult to change. It can also be difficult to change the type of an individual object, or insert records for objects whose type is uncertain. Iterative structuring or "sense-making" activities that are central to knowledge discovery are not easily supported by these traditional fixed schemas.

In contrast, PROXIMITY's schema facilitates a flexible approach to data representation. It allows for the transformation of the data as necessary throughout the discovery process. Instead of associating a fixed set of attributes with objects of a given type, attributes are each stored in a separate table. It is no longer assumed that any set of objects have the same structure, each object can be associated with any set of attributes. This approach escapes the rigid record typing of a traditional schema, and it enables analysts to introduce and transform data structures as analysis progresses.

Specifically, PROXIMITY's flexible schemas support a adaptable type system, facilitate attribute creation and allow for efficient scaling. The *type* of an object or link is an attribute like any other. An object or link can have no type, one type, or many types, and types of objects and links can be easily changed. New attributes can be added easily, without requiring the attribute be added to every object of a given type, even when values are not known. This type of representation also allows for fast access to groups of objects/links instead of fast access to full attribute information for a given object or relation. This trade-off of row-view for column-view allows for fast views of collections of heterogeneous subgraphs.

4. Use Many Types of Relational Features

In propositional datasets with sparse information, it can be hard to build accurate models. However, relational data may contain a wealth of information in the relationships even if there is only sparse attribute information for the objects. In many transactional datasets such as financial transactions or telephone call detail, the bulk of the data are contained in relations. In these situations little is known about the objects in isolation but it is still possible to build models using the transaction patterns. Fortunately, the relational structure provides a wealth of opportunities for construction of new features to use in building models.

4.1 Relational Features

We define *feature* as mapping between raw data and a low-level inference. For example, a feature in the gene data might be *Function=cellular organization*. In this case, the feature combines an attribute (function), an operator and a value. Typically, many features are combined into a higher-level model such as a decision tree or a rule set. Relational features are used by models that predict the value of an attribute based on the attributes of related objects. For example, we might use the localization of gene *y* to predict the function of gene *x* if *x* and *y* interact together. Relational features are similar to the features described above in that they identify both an attribute and a way of testing the values of the attribute. However, relation features may also identify a particular relation (e.g. *Interaction(x,y)*) that links a single object *x* to a set of related objects *Y*. If this is the case, the attribute referenced by the feature may belong to the related objects *Y* and the test is conducted on the set of attribute values in the objects in *Y*. For example, the relational feature:

$$Mode(Localization(Y)) = nucleus$$
$$where\ Gene(x), Y = \{y \mid Interaction(x, y)\}$$

determines whether the most prevalent localization of all the genes interacting with *x* is equal to *nucleus*.

When the relation between *x* and *Y* is one-to-many, a relational feature must considers a set of attribute values on the objects *Y*. In this situation, standard database aggregation functions (e.g. *max*, *mode*, *average*) can be used to map sets of values into single values. For example, if a gene' s location depends on the chromosome values of linked genes, and a gene interacts with five other genes, then a model could aggregate the five values of chromosome with a function such as *mode*. An alternative approach is to use first-order features that produce boolean values characterizing the neighborhood. For example, a first-order feature might determine whether another gene with a particular function and location interacted with a given gene. The attribute would be true if one or more genes met the function and location criteria.

4.2 Relational Feature Space

The size of the potential feature space for relational data is enormous, as is evidenced by the search space considered in many ILP systems. Flach and Lavrac have outlined a framework for first-order features (2000) but the treatment is limited to features consisting of conjunctions of literals. Much of the work in relational learning outside the ILP community has considered features using the ag-

gregation functions described in the previous section (e.g., Friedman et. al. 1999). Our own work for the KDD Cup consisted of creating over a hundred aggregated relational features. However, beyond aggregation of attribute values, there is a wealth of potential features concerning the structure of the data itself. Figure 3 outlines a range of feature types made possible by relational data. The x-axis represents the scope of the feature. *Individual* refers to features about objects in isolation. *Local* refers to features constructed from the local relational neighborhood on objects. *Global* refers to features constructed from the entire graph of objects. Individual features can only use attributes because an object in isolation has no graph structure. However, both local and global features can consider either graph structure or item attributes, or both.

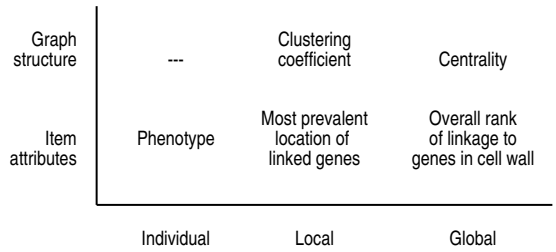

Figure 3: Feature space graphic

Figure 3 presents examples of features in each of the five types. *Phenotype* is an intrinsic attribute of genes that is provided in the raw data. Clustering coefficient is defined as the ratio of the number of linked neighbors (those who also link to each other) to the possible number of linked neighbors (Watts 1999). This measures the connectedness of surrounding genes. *Most-prevalent-location* was described in section 4.1; it considers the most prevalent localization of all the genes interacting with a given gene. Centrality measures help to determine the relative importance of nodes in the graph (Wassermann and Faust 1994). Betweenness centrality counts the number of shortest paths in the graph that travel through each gene. *Overall-rank-of-cell-wall-linkage* is a two-step feature. First the degree for each gene is measured with respect to genes located in the cell wall. Then all the genes receive a ranked according to these degrees. The highest ranked gene is the one that interacts with more genes located in the cell wall than any other gene in the genome.

Most, if not all of the features considered by current relational data mining systems fall into the categories of *Individual* and *Local Item-Attributes*. Structural features such as clustering coefficient and centrality are more common in social network analysis, and other algorithms such as Kleinberg's hubs and authorities algorithm (Kleinberg 1999) that have been developed in the computer science community have only begun to be explored. Kleinberg's algorithm was developed to quantify two measures of web page "interestingness." Hubs and authorities are defined recursively – a web page is an authority if it is linked to by many hubs, and it is a hub if it provides links to many authorities. The algorithm iteratively calculates hub and authority weights on all pages simultaneously using the link structure. Features such as these require a global view of the graph to be calculated. Features of this type have yet to be exploited by the community, and are available only if you retain a relational representation. Much work has been done in traditional machine learning to determine the utility of features and select the most useful. Both this work and work in automatic feature construction, are important to relational knowledge discovery systems because the number of potential relational features.

5. Autocorrelation is Ubiquitous

Our analysis indicates that *relational autocorrelation* is a common characteristic of relational data[2]. Informally, relational autocorrelation occurs when the values of a given attribute are highly uniform among objects that share a common neighbor. The fragment of the gene data in Figure 2 shows how many linked genes have highly correlated functions and locations.

We will define relational correlation *C(X,f,P,Y,g)* with respect to two sets of objects *X* and *Y*, two attributes *f* and *g* on objects in *X* and *Y*, respectively, and a set of paths *P* that connect objects in *X* and *Y*.

Definition: *Relational correlation C* is the correlation between all pairs *(f(x),g(y))* linked by paths in *P*. ■

Given the pairs of values that these elements define, traditional measures such as information gain, chi-square, and Pearson's contingency coefficient can be used to assess the correlation between values of the attributes *f* and *g* on objects connected by paths in *P*. The range of *C* depends on the measure of correlation used.

We can use the definition of relational correlation *C(X,f,P,Y,g)* to define relational autocorrelation as the correlation between the same attribute on distinct objects belonging to the same set.

Definition: *Relational autocorrelation C' is:*

$$C'(X, f, P) \equiv C(X, f, P, X, f) \text{ for } P = \{p(x_i, x_j) \mid x_i \neq x_j\}$$ ■

For example, *C'* could be defined with respect to gene objects, the attribute *localization* on genes, and paths formed by traversing *Interaction* links that connect the genes to other genes.

5.1 Autocorrelation in Relational Datasets

If the underlying measure of correlation varies between zero and one, then *C'=1* indicates that the value of the

[2] Much of this discussion is drawn from earlier papers (Jensen and Neville 2002a,b).

attribute for a specific node x_i is always equal to all other nodes x_j reachable by a path in P. When $C'=0$, values of $f(X)$ are independent. Table 1 gives estimates of relational autocorrelation for gene function and localization, linked through other genes. For a measure of correlation, Table 3 uses Pearson's corrected contingency coefficient (Sachs 1992), a measure that produces an easily interpreted value between zero and one. Autocorrelation is fairly strong for both class labels.

Table 1: Autocorrelation in the gene data

Autocorrelation Type	Value
C'(Function,Gene,Interaction)	0.52
C'(Localization,Gene,Interaction)	0.76

In addition to the gene data, we have encountered many other examples of high relational autocorrelation. For example, in our study of publicly traded companies, we found that when persons served as officers or directors of multiple companies, the companies were often in the same industry. Similarly, in data from the Internet Movie Database, movies linked through a common studio, director or producer have highly autocorrelated box-office receipts. Such autocorrelation has been identified in other domains as well. For example, fraud in mobile phone networks has been found to be highly autocorrelated (Cortes et al. 2001). The topics of authoritative web pages are highly autocorrelated when linked through directory pages that serve as "hubs" (Kleinberg 1999). Similarly, the topics of articles in the scientific literature are often highly autocorrelated when linked through review articles.

Relational autocorrelation represents an extremely important type of knowledge about relational data, one that is just beginning to be explored and exploited for learning statistical models of relational data (Neville and Jensen 2000; Slattery and Mitchell 2000). Deterministic models representing the extreme form of relational autocorrelation have been learned for years by ILP systems. By representing and using relational autocorrelation, statistical models can make use of both partially labeled data sets and high-confidence inferences about the class labels of some nodes to increase the confidence with which inferences can be made about nearby nodes.

Relational autocorrelation can also complicate learning and evaluating relational models. These issues are outlined further in section 7.

6. Transformative learning

Successful knowledge discovery is often an iterative process during which analysts coevolve their domain knowledge and the way in which they represent data about that domain. Building models of raw data is frequently unsuccessful but this phase (as well as subsequent phases) can suggest new ways to view the data, either by modification or by constructions of new features. Knowledge discovery can also be approached as a hierarchical process where simpler concepts need to be learned first and then used together to model more complex concepts. Supporting an iterative approach to discovery is useful for systems designed to analyze propositional data, and it is vital for systems designed to analyze relational data.

6.1 Supportive Architecture

We have found that the discovery process is intimately connected to how data are represented. Sometimes it is hard to know the 'right' way to represent relational data for analysis. For example, should a financial transaction between two people be represented as a 'transaction' object with relations to the two people or just as a relation between the two people? It could depend on the task. If the user is trying to model fraudulent businesses, then the first representation might be more useful to look at all transactions conducted at a particular site. However, if the user is trying to model fraudulent people, the second representation might be more useful to view aggregate behavior of an individual. Often the task is only to model 'fraud' and it is during the process of discovery that one or the other view becomes the focus of the analysis. In this situation, a hard and fast choice of representation up front could not only slow down discovery but also prevent it completely.

We have designed PROXIMITY with an architecture suited to managing structured data and methods that can both support and direct effective data transformations. In addition to the simple graph representation described earlier, PROXIMITY provides views of the graph using collections of subgraphs. For example, in the gene data we could create a collection of all the genes that localize in the cell wall and then analyze the properties of these genes to try to predict function. This would be useful if we were able to build an accurate model of localization that could be applied to test data first and then have a second model that predicts function given localizations.

We could also create a collection of subgraphs in the gene data to view each object in its local relational neighborhood. In this case, each subgraph in the collection would consist of a core gene object and there would be a separate subgraph for each gene in the data. Each subgraph could contain all the other genes that the core gene interacts with; these genes may be duplicated across subgraphs. This approach allows for fast calculation of features concerning the local neighborhood of a gene. Most of the attributes we constructed for the KDD Cup were calculated using such a view of the gene data. The view made it simple to calculate such attributes as 'count of linked genes in golgi' and 'sum of interaction expressions to linked genes whose function is cellular transport.'

Views of the graph facilitate the analysis process by allowing an analyst to filter and abstract the data. Filtering the data is often useful in data sets with large numbers of

instances or attributes, but it can also be useful when the data have many types of objects or links as well. Often, filter the data can reveal new associations that were previously hidden or swamped out by other portions of the data. Our analysis of the gene data included calculating autocorrelations for genes linked in various ways. For, example we filtered the links by type and expression. We also looked at pairs of interacting genes that shared the same function or location. Table 2 reports several of the high autocorrelations we found using filtered links.

Table 2: Autocorrelation in the gene data

Autocorrelation Type	Value
C'(Func=transcription,Gene,Interact-same-loc)	0.92
C'(Func=transport,Gene,Interact-type=gen-phys)	0.96
C'(Loc=nucleus,Gene,Interact-type=genetic)	0.84
C'(Loc=golgi,Gene,Interact-same-func)	0.62

6.2 Inductive Closure

The ability to transform the data easily and repeatedly throughout the discovery process is important not only to improve efficiency but it is also crucial to evolving knowledge about the domain. To this end, we designed a system with *inductive closure*. This denotes that the system is closed under induction -- the results of any analysis feed directly back into the system to be reused by further analysis. This design choice is again motivated by the iterative nature of the discovery process and a view of the database as a blackboard. If relationships in the data are to be used effectively to improve the quality of predictions, a system's model output should feed directly into the next model-building phase.

PROXIMITY modules use a small number of special purpose data structures to make interaction between modules possible and is accompanied with a scripting language to support dynamic compositions by analysts using any number of analysis methods. PROXIMITY classifiers operate on collections, use attributes and produce attributes to record predictions. PROXIMITY graph queries take a subgraph specification, output collections of matches and facilitate adding new object/links and or calculating attributes based on the information in the subgraphs. PROXIMITY graph calculation modules that construct views such as connected components or paths operate on collections and produce either collections or attributes.

7. Statistical Opportunities and Challenges

Relational data have the potential to drastically improve not only the accuracy of learned models, but also the quality of discovered patterns. However, the dependence among instances in the data introduces new statistical challenges and opportunities for relational learning algorithms. As we seek to represent and use relational autocorrelation in statistical models of relational data, we will need to adjust for its effects on feature selection and evaluation. Solving these issues could result in huge benefits by reducing the sample complexity of the learning algorithms and increasing the accuracy of the learned models.

7.1 Feature Selection Bias

Two common characteristics of relational data sets — *concentrated linkage* and *relational autocorrelation* — can cause learning algorithms to be strongly biased toward certain features, irrespective of their predictive power. In a related paper (Jensen and Neville 2002a) we show how dependence among the values of a class label in relational data can complicate feature selection in methods for machine learning. We show how linkage and autocorrelation can combine to reduce the *effective sample size* of some data sets, introduce additional variance, and lead to feature selection bias. To our knowledge, no current relational learning algorithm accounts for this bias. Resampling can be used to obtain accurate estimates of variance for each feature and we are currently investigating techniques to use those estimates to improve feature selection in relational data.

7.2 Bias in Evaluation

Linkage and autocorrelation can also cause traditional methods of evaluation to greatly overestimate the accuracy of induced models on test sets. Accurate evaluation of learning algorithms is central to successful research in relational learning. The most common method for evaluating a learning algorithm is to partition a given data sample into training and test sets, construct a model on the training set and evaluate the accuracy of that model on the test set. Dependence among the values of a class label in relational data can cause strong biases in the estimated accuracy of learned models when accuracy is measured in this way. In general, current techniques for evaluating relational learning algorithms do not account for this bias. We've presented a new sampling algorithm that can be applied to any relational data set to eliminate this bias in a paper submitted elsewhere (Jensen and Neville 2002b). This approach removes the dependence between training and test sets by confining any correlation among class labels to a single subsample (i.e. within the subsamples instead of across the subsamples).

7.3 Improving Predictions

In addition, maintaining relational representations allows inference procedures to exploit relational autocorrelation to improve the predictive accuracy of models. By preserving the relational structure of the data, we can exploit the connections between objects to improve classification accuracy. We have developed an *iterative classification* technique (Neville and Jensen 2000) based on the premise that if two target objects are related, inferring the class of

one can tell us something about the class of the other. As the name suggests, iterative classification runs a classifier many times on the same collection of subgraphs, recalculating attribute values after each iteration. Inferences made with high confidence in initial iterations of the classifier are fed back into the data to strengthen inferences about related objects in subsequent iterations.

8. Conclusions

The biases associated with linkage and autocorrelation indicate the importance of maintaining relational data representations, rather than propositionalizing data. Maintaining a relational data representation makes it possible to assess the statistical effects of linkage and autocorrelation, and to adjust for the resulting bias. In addition, relational representations allow inference procedures to exploit relational autocorrelation to improve the predictive accuracy of models. Finally, relational representations extend the space of potential relational features; in particular, global and structural features present a set of previously unexplored features.

A co-evolutionary approach to knowledge formation and data transformation addresses the "chicken and egg" character of knowledge discovery. Knowing the "right" representation can be crucial to the discovery process but often this knowledge is not known up front and needs to be learned during analysis. The additional information contained in relationships allows for alternate views of the data and as such contain many possibilities for transformation during learning. Systems designed to support an iterative discovery approach promise to clarify the utility of various representations and will enable dramatic improvements in knowledge discovery processes.

Acknowledgements

This paper would not have been possible without the collaborative environment within the Knowledge Discovery Laboratory. In particular, the authors thank Matt Cornell and Hannah Blau for their contributions to the PROXIMITY project over the past three years. This research is supported under a National Science Foundation Graduate Research Fellowship and by DARPA and AFRL, AFMC, USAF under contract numbers F30602-00-2-0597 and F30602-01-2-0566. The U.S. Government is authorized to reproduce and distribute reprints for governmental purposes notwithstanding any copyright notation hereon. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements either expressed or implied, of the DARPA, the AFRL or the U.S. Government.

References

Cheng, J., C. Hatzis, H. Hayashi, M. Krogel, S. Morishita, D. Page, J. Sese. (2002). KDD Cup 2001 Report. SIGKDD Explorations.

Cortes, C., D. Pregibon, and C. Volinsky (2001). Communities of Interest. Proceedings Intelligent Data Analysis 2001.

Dzeroski, S. and N. Lavrac, (Eds.) (2001). Relational Data Mining. Berlin: Springer.

Flach, P. and N. Lavrac. (2000). The role of feature construction in inductive rule learning. Proc. of ICML2000 Workshop on Attribute-value and Relational Learning.

Friedman, N., L. Getoor, D. Koller, and A. Pfeffer (1999). Learning Probabilistic Relational Models. In *IJCAI'99*. 1300-1309.

Jensen, D. (1998). Statistical Challenges to Inductive Inference in Linked Data. Proceedings of the 7th International Workshop on Artificial Intelligence and Statistics.

Jensen, D. and J. Neville (2002a). Linkage and Autocorrelation cause bias in relational feature selection. Proc. 19th International Conference on Machine Learning.

Jensen, D. and J. Neville (2002b). Autocorrelation and linkage cause bias in evaluation of relational learners. Submitted: 12th International Conference on ILP.

Kleinberg, J. (1999). Authoritative sources in a hyperlinked environment. Journal of the ACM 46:604-632,

Neville, J. and D. Jensen (2000). Iterative Classification in Relational Data. AAAI Workshop on Learning Statistical Models from Relational Data, 42-49.

Sachs, L. (1982). Applied Statistics. Springer-Verlag.

Silverstein, G. and Pazzani, M. (1991). Relational cliches: Constraining constructive induction during relational learning. Proceedings of the Eighth International Workshop on Machine Learning.

Slattery, S., and Mitchell, T. (2000). Discovering test set regularities in relational domains. Seventeenth International Conference on Machine Learning.

Wassermann, S. and Faust, K (1994). Social Network Analysis: Methods and Applications, Cambridge: Cambridge University Press.

Watts, D. (1999). Small Worlds. Princeton Univ. Press.