Simple Decision Forests for Multi-Relational Classification

Bahareh Bina, Oliver Schulte, Branden Crawford, Zhensong Qian, Yi Xiong

School of Computing Science, Simon Fraser University, Burnaby, B.C., Canada V5A 1S6

Abstract

An important task in multi-relational data mining is link-based classification which takes advantage of attributes of links and linked entities, to predict the class label. The relational naive Bayes classifier exploits independence assumptions to achieve scalability. We introduce a weaker independence assumption to the effect that information from different data tables is independent given the class label. The independence assumption entails a closed-form formula for combining probabilistic predictions based on decision trees learned on different database tables. Logistic regression learns different weights for information from different tables and prunes irrelevant tables. In experiments, learning was very fast with competitive accuracy.
Keywords: link-based classification, multi-relational Naive Bayes classifiers, multi-relational decision trees, logistic regression

1. Introduction

Most real-world structured data are stored in the relational format, with different types of entities and information about their attributes and links between the entities. Relational data classification is the problem of predicting a *class label* of a target entity given information about features (attributes) of the entity, of the related entities, and of the links. One of the issues that makes link-based classification difficult compared to single-table learning is

Email addresses: bba18@cs.sfu.ca (Bahareh Bina), oschulte@cs.sfu.ca (Oliver Schulte), bsc7@sfu.ca (Branden Crawford), zqian@sfu.ca (Zhensong Qian), yi_xiong@sfu.ca (Yi Xiong)

the large number of different types of dependencies that a model may have to consider [1, 2]. A principled way to approach the complexity of correlation types is to consider model classes with explicitly stated independence assumptions. The aim is to make a good trade-off between the expressive power of the model on the one hand, and the scalability of learning on the other. Multi-relational Naive Bayes net classifiers (NBCs) are a prominent example of this approach [3, 4, 5]. Naive Bayes Classifiers incorporate two different kinds of independence assumptions: (1) *cross-table independence*, roughly that information from different tables is independent given the target class label, and (2) *within-table independence*, that descriptive attributes from the same table are independent given the target class label. The approach of this paper is to maintain the first assumption, but to drop the second one. This allows us to capture and exploit complex dependencies among the attributes and the class label within each table.

The Classification Model. The target table is the table that contains the class attribute. Consider the set of tables that can be joined to the target table via a chain of foreign key links; we refer to the corresponding table joins as join tables. We define a *cross-table Naive Bayes assumption*, according to which different join tables are independent given the class label, and mathematically derive from it a novel log-linear classification formula. We extend the log-linear classification model to allow different join tables to contribute more or less strongly to the classification decision, with weights for the relative contributions learned by a logistic regression algorithm. Zero weights can be used to prune irrelevant tables.

Simple Decision Forests. We use a decision tree classifier as a base learner for dependencies within a single table, for the following reasons. (1) Decision tree learners are very fast. (2) There are well-researched methods for learning trees that provide class probability estimates in their leaves [6, 7, 8, 9]. Such trees are sometimes called *probability estimation trees*; in this paper, we use the simpler term decision tree. (3) Other multi-relational decision tree classifiers are available for comparison. (4) Decision trees easily handle continuous and categorical variables. (5) Decision trees support rule learning, and conversion to logic-based relational models, such as Markov Logic Networks [10]. (6) Decision trees are one of the most popular classifiers. Brydon and Gemino describe applications of decision trees in decision support and business intelligence [11]. An empirical comparison shows a large run-time

advantage and a strong predictive performance, that is better than that of previous relational decision tree learners, and of a relational Naive Bayes classifier.

Contributions. The main novel contributions of our work are as follows.

1. Use of naive Bayes assumption across tables but not within tables.
2. Use of logistic regression to assign weights for the contributions from different linked tables, including zero weights for pruning.
3. A new method, based on independence assumptions, for upgrading a single-table decision tree classifier for multi-relational classification.
4. An experimental evaluation on five real life databases, demonstrating that our method is very fast with good predictive accuracy.

Paper Organization. We first describe related work, review background material and define our notation. Then we formalize the independence assumptions for multi-relational learning and derive the classification formula. We describe the simple forest induction, use of logistic regression, and classification algorithms. The final section evaluates the classification performance of the different models on three benchmark datasets.

2. Related Work

We selectively describe relevant multi-relational classification approaches.

Feature Generation. Most of the work on relational upgrades of single-table classifiers uses a mechanism for relational feature generation. Feature generation is done by using aggregate functions to summarize the information in links [12, 13, 14, 15, 16]. Inductive Logic Programming (ILP) approaches use existentially quantified logical rules [17, 18]. Several recent systems combine both aggregation and logical conditions (e.g., [12, 15]). Treeliker is a recent propositionalization system that uses monotonicity properties to generate non-redundant conjunctive features, with state-of-the-art efficiency performance [16]. The generated features can be used by any propositional learner. The predictors in our model are not derived features, but the descriptive attributes as defined in the relational database schema. Relational feature generation is by far the most computationally demanding part of such approaches. (For an example, generating 100,000 features on the CiteSeer dataset, which is smaller than the databases we consider in this paper, can

take several CPU days [15, Ch.16.1.2]). Models that use independence assumptions rather than generated features to combine the information from different tables are orders of magnitude faster to learn, as our experiments confirm. At the end of the paper we discuss how our approach can be extended to support aggregate feature generation.

Relational Decision Trees. To our knowledge, all previous work on relational decision trees has followed the propositionalization/aggregation paradigm (e.g., [12, 19, 14]). We do not utilize aggregate functions or existential quantification (which may be viewed as an aggregation mechanism). Instead, we consider each link independently and utilize all the information from them. Using aggregation/quantification implies loss of information [4]. FORF (First Order Random Forest) learns an ensemble of different decision trees, each of which is constructed using a random subset of the schema features [12]. Nodes of the trees are first-order logic queries. First order random forests can be categorized according to different levels of aggregation. FORF-NA is the simplest one with no aggregation function. FORF-SA uses simple aggregation, and FORF-RA employs refinements of aggregate queries. The final decision is made by averaging the results of trees. In contrast, in our model we learn different trees using all attributes of each table. Attributes within a table are typically more correlated than attributes from different tables, so it is advantageous to learn trees from each table's attributes jointly than on random subsets of features. We combine the results using the weights learned by logistic regression rather than plain averaging.

TILDE (Top-down induction of first-order logical decision tree) is a prominent relational decision tree within the Inductive Logic Programming framework [19]. The nodes in TILDE trees test first-order conditions. The trees are learned with a divide and conquer algorithm similar to C4.5 [20]. The main difference with our model is that Tilde uses the existential quantifier to build rules. Runtime efficiency is a challenge in Tilde, because it potentially generates many rules to find the best ones.

MRDT (Multi Relational Decision Tree) constructs the decision tree by using selection graphs as the nodes of the tree [21]. A selection graph is a directed graph which imposes a set of constraints on incorporating information from several tables. MRDT use the existential quantifier to aggregate information from different links. Guo et al in [22] speed up the MRDT algorithm by using id propagation to implement a virtual join operation that avoids the cost of physical joins.

Independence Assumptions. There are various proposals to apply Naive Bayes (NB) Assumptions and logistic regression (LR) to relational data. Logistic regression can be viewed as a discriminative version of the generative Naive Bayes Classifier model [23]. For single-table classification, the advantages of logistic regression over simple Naive Bayes Classifier have been studied in detail [23], and similar results have been reported for single-relation classification [24]. Popescul and Unger combine logistic regression with an expressive feature generation language based on SQL queries [15].

Much of the work on relational Naive Bayes Classifiers combines first-order features defined by logical rules [25, 26, 17]. The nFOIL system utilizes the Naive Bayes assumption to learn existentially quantified first-order features. Neville *et al.* [3] investigate different functions for aggregating the influence of linked objects. Chen *et al.* discuss the pros and cons of using aggregation vs. independence assumptions, and argue that for a relational Naive Bayes Classifier, the use of aggregate function loses statistical information and is not necessary [4]. The multi-relational Naive Bayes Classifier assumes that all attributes in a database are independent, including attributes from the same table [4]. We use a weaker assumption by learning decision trees for attributes within the tables.

CrossMine [18] is an efficient ILP-style rule learner algorithm. It uses tuple ID propagation to virtually join tables, and randomly selects instances to overcome the skewness of datasets.

Structured logistic regression treats the influence on the class label of the target table and that of linked tables as independent [24]. Two independent logistic regressions are carried out, one on the target table and the other on the set of linked tables. Information from links is summarized using aggregate functions. The product of the results determines the class label. In our method we use a stronger assumption that links are independent of each other, and learn a model for each link separately. The predictors in our model are probabilities predicted based on local information rather than aggregates as in structured logistic regression.

The Heterogeneous Naive Bayes classifier [5] considers join tables conditionally independent, and combines the posterior results from the table classifiers applied separately to linked and target tables by a naive Bayes assumption. A key difference is the classification model: We formally derive our classification formula, which normalizes the posterior probability from each linked table by the class prior. Also, we use decision trees, and we employ logistic regression to combine the contributions of different linked

tables.

Ensemble Classifiers. Our model has some resemblance to ensemble/committee classifiers in that we combine the probabilistic outputs of different classifiers. The main differences are as follows.

1. Most ensemble methods grow the set of classifiers, e.g., a decision forest, *dynamically* in the course of learning. In a simple decision forest, the set of decision trees is *fixed by the database schema*, since each tree represents a classifier for a given relational pathway (chain of foreign key links).
2. In many ensemble models, gating functions are learned that assign different classifiers to different parts of the input feature space. In our simple decision forest, the partition of the input feature space is predetermined by the database schema.
3. There are potentially many methods for combining the predictions of a classifier ensemble [27, 28]. Our cross-table Naive Bayes independence assumption entails a log-linear combining method.

Graphical Models. Ideas from directed graphical models are used by Probabilistic Relational Models [29], which learn a Bayesian network model of the data distribution, and use aggregate functions to specify conditional probability tables.

Unlike directed graphical models which impose an acyclicity constraint, undirected ones do not have a cyclicity problem and are widely used for LBC [30, 10, 31]. Two major formalisms are relational conditional Markov random fields [30] and Markov Logic Networks (MLNs) [10]. Relational Markov random fields are a general log-linear model that does not use aggregate functions nor independence assumptions [10]. The classification formula we derive from the cross-table Naive Bayes assumption is also a log-linear model, which shows that our model is a *subclass* of relational conditional random fields. This observation provides a probabilistic interpretation of our classification formulas in terms of a general graphical model. While compared to general Markov models, the expressive power of our log-linear model is restricted by independence assumptions, the trade-off for the expressive power of more general log-linear models is higher complexity in learning; various studies have shown that scaleable learning is a major challenge for Markov model learning in the multi-relational setting [32, 33, 31].

In practice one often needs to perform collective classification: predict the class label of several interrelated entities simultaneously. While collective classification has received much attention [30, 34], our theoretical framework for multi-relational classification formulas is already quite rich, so our empirical evaluation focuses only on individual classification and leaves applications to collective classification for future work. A principled approach to collective classification is to convert decision forests to Markov Logic Networks, as described in Section 3.2, and then apply MLN techniques for collective classification [10].

3. Preliminaries and Notation

A standard **relational schema** contains a set of tables. A **database instance** specifies the tuples contained in the tables of a given database schema. If the schema is derived from an entity-relationship model [35, Ch.2.2], the tables in the relational schema can be divided into *entity tables* and *relationship tables.* Relationship tables link Entity tables to each other by foreign key pointers. The **natural join** of two tables is the set of tuples from their cross product that agree on the values of fields common to both tables.

3.1. Pathways and Join Tables

One of the key challenges in multi-relational classification is the multiplicity of pathways or table joins through which the target entity may be linked to other entities. Han et al. [4] proposed a graphical way to structure the space of possible pathways. A **Semantic Relationship Graph** (SRG) for a database $\mathcal{D}$ is a directed acyclic graph (DAG) whose nodes are database tables in $\mathcal{D}$ and whose only source (starting point) is the **target table** T. For simplicity we assume that the target table is an entity table. If an edge links two tables in the Semantic Relationship Graph, then the two tables share at least one primary key. For each path $T, T_1, \ldots, T_k$ in an Semantic Relationship Graph, there is a corresponding valid join $T \bowtie T_1 \cdots \bowtie T_k$. Because the number of attributes in a valid join may become quite large, Han et al. use only the attributes of the last table in the path, and the class attribute from the target table, which is called *propagating* the class attribute along the join path [4]. An **extended database** is a database that contains all valid join tables with selecting (projecting) the class attribute and attributes from the last table in each join. We refer to the tables in the extended database as **e**xtended join tables, or simply join tables. A **decision**

Figure 1: Semantic relationship graph for the university schema. Each path in the Semantic Relationship Graph corresponds to a join table in the extended database.

tree for an extended join table contains nodes labelled with attributes from the join table. The leaves contain a probability distribution over the class labels. A **simple decision forest** contains a decision tree for each join table in the extended database.

Example. A university schema is our running example. The Semantic Relationship Graph is shown in Figure 1. The schema has three entity tables: *Student*, *Course* and *Professor*, and two relationships: *Registration* records *Courses* taken by each *Student* and *RA* records research assistantship of students for professors. An instance for this schema is given in Figure 2. The class attribute is *Intelligence* of a *Student*. Therefore, in the Semantic Relationship Graph we have *Student* table as the source node. Figure 3 shows an extended university database instance that results from propagating the class attribute *Intelligence*. The paths in the Semantic Relationship Graph of Figure 1 correspond to join tables in Figure 3. In the extended university database valid joins include the following:

Course

c-id	Rating	Difficulty
101	3	1
102	2	2

Student

s-id	Intelligence	Ranking
Jack	?	1
Kim	2	1
Paul	1	2

Professor

p-id	Popularity	Teaching-a
Oliver	3	1
Jim	2	1

Registration

s-id	c.id	Grade	Satisfaction
Jack	101	A	1
Jack	102	B	2
Kim	102	A	1
Paul	101	B	1

RA

s-id	p-id	Salary	Capability
Jack	Oliver	High	3
Kim	Oliver	Low	1
Paul	Jim	Med	2

Figure 2: A small instance of a university database.

- *Student* $\bowtie$ *Registration* with (a projection of) the attributes from *Registration*, and the class label from *Student* (e.g., Intelligence of Student).
- *Student* $\bowtie$ *Registration* $\bowtie$ *Course*, with (a projection of) the attributes from *Course*, and the class label from *Student*.
- *Student* $\bowtie$ *RA*, with (a projection of) the attributes from *RA*, and the class label from *Student*.
- *Student* $\bowtie$ *RA* $\bowtie$ *Professor*, with a selection of the attributes from *Professor*, and the class label from *Student*.

Figure 4 shows a simple decision tree for the extended join table *Student* $\bowtie$ *RA* $\bowtie$ *Prof* table that may be learned from data like that shown in Figure 1.

Student

s-id	Intelligence	Ranking
Jack	?	1
Kim	2	1
Paul	1	2

Student∞Registration∞Course

s-id	c.id	Rating	Diff	Intelligence
Jack	101	3	1	?
Jack	102	2	2	?
Kim	102	2	2	2
Paul	101	3	1	1

RA

s-id	p-id	Salary	Capability	Intelligence
Jack	Oliver	High	3	?
Kim	Oliver	Low	1	2
Paul	Jim	Med	2	1

Student∞RA∞Prof

s-id	p-id	Popularity	Teaching-a	Intelligence
Jack	Oliver	3	1	?
Kim	Oliver	3	1	1
Paul	Jim	2	1	2

Registration

s-id	c.id	Grade	Satisfaction	Intelligence
Jack	101	A	1	?
Jack	102	B	2	?
Kim	102	A	1	2
Paul	101	B	1	1

Figure 3: An instance of the extended university database. The target entity is Jack, the target table is *Student*, and the class label is *Intelligence*.

3.2. Model Conversions

We discuss how decision forests can be converted to other model types. If knowledge discovery is the goal of data analysis, the rule format provides an accessible presentation of statistical regularities for users. Simple decision forests support *rule extraction* because each join table corresponds to an Semantic Relationship Graph path. For each table decision tree, each branch converts to a rule in the usual way (conjunction of conditions along the

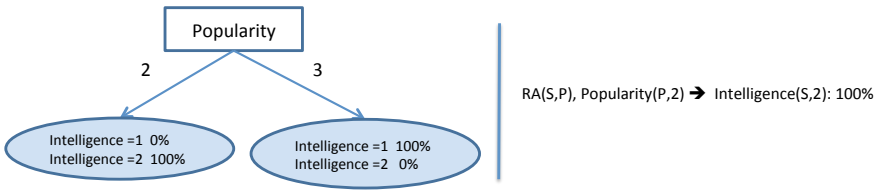

Figure 4: Left: A simple single decision tree that may be learned for data in the format of the extend join table *Student* $\bowtie$ *RA* $\bowtie$ *Prof* table. Right: A logical rule extracted from the tree, corresponding to the leftmost branch.

branch). The relational context in which the rule holds is specified by the join conditions of the table. Figure 4 (right) provides an example.

Markov Logic Networks (MLNs) are a prominent statistical-relational model class that combine first-order logic with Markov random fields (undirected graphical models) [10]. An MLN comprises a set of weighted first-order clauses. In terms of graphical models, an MLN is a template for a Markov random field, with a log-linear likelihood function that is the weighted sum of counts of features defined by the first-order formulas. A state-of-the-art approach to learning the clauses in an MLN is to first learn a set of decision trees and then convert each branch of each decision tree to an MLN clause [36, 33]. The weights of the clauses are obtained from probability estimation trees by using the log-conditional probabilities associated with a leaf [36], and from regression trees by using the regression weights [33]. The example rule from Figure 4 (right) would induce the MLN clause

$$RA(S,P), TeachingA(P,1), Intelligence(S,2) : w = ln(100\%).$$

For general discussion and details on MLNs and the conversion from decision forests to MLNs please see [36, 33]. The conversion to Markov Logic Networks provides a probabilistic foundation for our classification approach in terms of log-linear models, and offers a principled approach to collective classification with the cross-table Naive Bayes assumption (see Sec. 2).

4. The Cross-Table Naive Bayes Assumption

In this section we define our independence assumption formally and derive a classification formula. The formula entails a logistic regression model for multi-relational classification. Applying the regression to decision trees defines the simple decision forest classification model.

4.1. Independence Assumptions

In the definitions below, we view a table M_i as a conjunction of the information pieces in it, that is, as a conjunction of value assignments . We write $M_{i,r}$ for the tuple of values in row r of table M_i without the primary key(s). For example, suppose that M_i is the *Student* $\bowtie$ *Registration* $\bowtie$ *Course* table from Figure 3. Then $M_{i,1} = \langle 3, 1, ? \rangle$ and $M_{i,3} = \langle 2, 2, 2 \rangle$.

Definition 1. *Consider an extended database with target table T, containing class attribute c and non-class attributes $\boldsymbol{a}$, and join tables $J_1, \ldots, J_m$. The* ***Inter-Table Independence Principle*** *states that*

$$P(T, J_1, \ldots, J_m|c) = P(\boldsymbol{a}|c) \prod_{i=1}^{m} \prod_{r=1}^{rows_i} P(J_{i,r}|c). \tag{1}$$

For instance, we may have $t = jack$, $c = Intelligence$ and $\boldsymbol{a} = \{Ranking\}$, so $\boldsymbol{a}(jack) = \langle 1 \rangle$.

Discussion. Equation (1) says that, conditional on the class label, the probability of the extended database instance is the product of the probability of the target attributes with the probabilities of each row of each join table. This assumption combines two principles, which we separate for discussion. First, that the information in different tables is independent given the class label:

$$P(T, J_1, \ldots, J_m|c) = P(\boldsymbol{a}|c) \prod_{i=1}^{m} P(J_i|c) \tag{2}$$

and second, that the information in different rows is independent given the class label, for each table i with $rows_i$ rows:

$$P(J_i|c) = \prod_{r=1}^{rows_i} P(J_{i,r}|c). \tag{3}$$

It is easy to see that Assumptions (2) and (3) entail the Inter-Table Independence Principle (1). Figure 5 illustrates Assumption (2) for the university schema using a Bayesian network. Since the class attribute is propagated to all join tables, table independence requires conditioning on this common information. This is an instance of the general principle that structured objects may become independent if we condition on their shared components. For instance, in Figure 3 the join tables Student-Registration-Course and

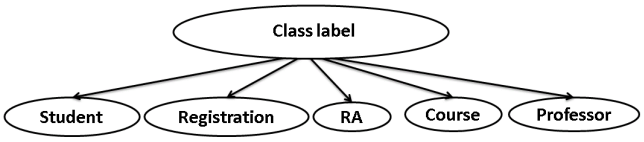

Figure 5: An example of the Table Independence Principle in the university domain. Attributes of different tables extended by the class label are independent of each other given the class label.

Student-RA-Prof both share the information about the intelligence of students.

Assumption (3) says that, given the common class label, rows of the link table join are independent of each other. This assumption is required to apply a single table classifier to each extended table. For instance, in Figure 3 two rows in the Registration table with the same student-id both share the information about the intelligence of the student.

Examples. Consider the extended university database from Figure 3 with the four join tables as shown. Then by Assumption (2), the joint probability of the database instance conditional on the class label factors as

$$\begin{aligned} P(T, J_1, J_2, J_3, J_4 | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &= \\ P(\mathit{Ranking}(\mathit{jack}) = 1 | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &\cdot \\ P(\mathit{Student} - \mathit{Registration} | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &\cdot \\ P(\mathit{Student} - \mathit{Registration} - \mathit{Course} | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &\cdot \\ P(\mathit{Student} - \mathit{RA} | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &\cdot \\ P(\mathit{Student} - \mathit{RA} - \mathit{Prof} | \mathit{Intelligence}(\mathit{jack}) = 2) & \end{aligned}$$

where the expression $\mathit{attribute}(\mathit{jack}) = 2$ denotes that the attribute value for target entity *jack* is 2.

Applying the Assumption (3) to the join table $J_i = \mathit{Student} - \mathit{Registration} - \mathit{Course}$ we have

$$\begin{aligned} P(\mathit{Student} - \mathit{Registration} - \mathit{Course} | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &= \\ P(\mathit{Rating}(101) = 3, \mathit{Diff}(101) = 1 | \mathit{Intelligence}(\mathit{jack}) = 2) \quad &\cdot \\ P(\mathit{Rating}(102) = 2, \mathit{Diff}(102) = 2 | \mathit{Intelligence}(\mathit{jack}) = 2) & \end{aligned}$$

where the expression $\mathit{attribute}(\mathit{course}) = x$ denotes that the attribute

value for course *course* is x. Rows that do not contain the target entity *jack* are not included in the equation.

Multi-relational Naive Bayes (NB) classifiers [4] add yet another assumption: the Column Independence Principle that within each row of each join table, the attributes are independent given the class label, which amounts to applying the single-table NB classifier in each join table. Using only Inter-Table Independence adds a degree of freedom that allows us to apply a classifier other than single-table NB to the join tables. Data from the same table is more likely to be correlated than data from different tables, because the database designer groups related attributes within a table. Thus by *not* assuming Column Independence, we exploit the domain knowledge implicit in the database design.

Impact of Assumption. We emphasize that we do not claim that the Inter-Table Independence Assumption is exactly true in a given database. For example, if we know that Jack is an intelligent student, the row independence principle (3) implies that his grades in course 101, and his grades in course 102, are independent of each other. There will in general be dependencies among different links/link attributes of the same entity. Therefore in a given dataset, these assumption may not be entirely but only approximately true. The use of relational independence principles is best viewed as a simplifying approximation to the actual dependencies in the dataset. Like the non-relational Naive Bayes assumption, the assumption permits accurate predictions of an entity's attributes even when false [37]. Another view is that it represents which correlations are modeled: Namely correlations between attributes and the class label given the link structure, but not correlations among the links or among the attributes of non-target entities. Analogously, the Naive Bayes assumption allows a model to present correlations between features and the class label in a single table, but not correlations among the features. As our experiments illustrate, the assumption leads to highly scalable learning, while it allows a model to capture the most relevant correlations from the data. To develop a learning algorithm and to investigate the impact of the assumption empirically, we derive a classification formula from it.

5. The Multi-Relational Classification Model

Let t denote the target entity from table T, such that $c(t)$ is the target class label and $\boldsymbol{a}(t)$ its non-class features. For simplicity we assume a binary

class label, so $c(t) \in \{0, 1\}$; our derivation can be extended to multi-class problems. Given a testing example (all the information in the extended tables) the posterior class odds are given by:

$$\frac{P(c(t) = 0|T, J_1, \ldots, J_m)}{P(c(t) = 1|T, J_1, \ldots, J_m)} = \frac{P(T, J_1, \ldots, J_m|c(t) = 0)P(c(t) = 0)}{P(T, J_1, \ldots, J_m|c(t) = 1)P(c(t) = 1)}$$

The class label is 0 if the posterior odds is larger than 1 and 1 otherwise. Applying the Inter-Table Independence Assumption (1), we have:

$$= \frac{P(c(t) = 0)P(\boldsymbol{a}(t)|c(t) = 0) \prod_{i=1}^{m} \prod_{r=1}^{rows_i} P(J_{i,r}|c(t) = 0)}{P(c(t) = 1)P(\boldsymbol{a}(t)|c(t) = 1) \prod_{i=1}^{m} \prod_{r=1}^{rows_i} P(J_{i,r}|c(t) = 1)}$$

where $rows_i$ is the number of rows in join table J_i. Substituting posterior probabilities using Bayes' theorem leads to the final **classification formula**:

$$= \frac{P(c(t) = 0|\boldsymbol{a}(t))}{P(c(t) = 1|\boldsymbol{a}(t))} \prod_{i=1}^{m} \prod_{r=1}^{rows_i} \frac{P(c(t) = 1)}{P(c(t) = 0)} \cdot \frac{P(c(t) = 0|J_{i,r})}{P(c(t) = 1|J_{i,r})} \tag{4}$$

5.1. The Weighted Log-Linear Classification Model

We convert the classification formula 4 to a parametrized log-linear model that assigns different weights to information from different database tables. Adding weight parameters and scale factors to Formula 4 leads to our final **log-linear classification model**:

$$\begin{aligned} \log\left(\frac{P(c = 0|T, J_1, \ldots, J_m}{P(c = 1|T, J_1, \ldots, J_m)}\right) = w_0 + w_T \log\left(\frac{P(c(t) = 0|\boldsymbol{a}(t))}{P(c(t) = 1|\boldsymbol{a}(t))}\right) + \\ \sum_{i=1}^{m} \frac{w_i}{rows_i} \sum_{r=1}^{rows_i} \log\left(\frac{P(c(t) = 1)}{P(c(t) = 0)}\right) + \log\left(\frac{P(c(t) = 0|J_{i,r})}{P(c(t) = 1|J_{i,r})}\right) \end{aligned} \tag{5}$$

First, the weighted log of the posterior odds given the information in the target table is computed. Second, for each of the join tables and for each row in it, we compute the log of: the posterior odds, given the attributes of the join table, *divided by the prior class odds.* This quantity measures how much the information from the join table changes the prior probability of a class label. Thirdly, the log-contribution from each extended table is weighted and scaled by the number of its rows. Finally, the weighted log-contributions are added together to predict the class label. Algorithm 1 shows the classification algorithm that corresponds to Formula 5.

Motivation. A direct application of inter-table independence assigns the same weight for the information obtained from different sources. The weights $w_0, w_1, \ldots, w_m$ adaptively control the impact of information from different links for class prediction.[1]

If an object has a large number of links, the information from links contributes many more terms in Equation (5) than the target entity's attributes. For example if Jack has taken 6 courses, the information from attributes of Jack, like *ranking*, is overwhelmed by information about courses. Normalizing the total log-contribution from each table by its size puts the contribution from tables with different sizes on the same scale: This uses the *average* log-contribution from the rows in a table, rather than the *sum total* of the log-contributions over all rows in the table. Scaling factors have been previously used with log-linear models [10, 38].

Algorithm 1 Multi-Relational Data Classification

Input:
(1) A new target instance t.
(2) Extended data base tables $J_1, \ldots, J_k$.
(3) Regression weights $\vec{w}$.
(4) Probabilistic Classifier $\mathcal{C}_T$ for the target table, $\mathcal{C}_i$ for each extended table.
Output: A predicted class label

$TP := w_0 + w_1 \cdot (\log(\mathcal{C}_T(c = 0|a(t)) - \log(\mathcal{C}(c = 1|a(t)))$ {Posteriors of the target table}
for all (extended tables J_i) **do**
  $LP := 0$
  for each (row $J_{i,r}$ containing the target entity t) **do**
    $LP+ = \log\left(\frac{\mathcal{C}_i(c(t=0)|J_{i,r})\cdot\mathcal{C}_i(c(t)=1)}{\mathcal{C}_i(c(t=1)|J_{i,r})\cdot\mathcal{C}_i(c(t)=0)}\right)$
  end for
  $TP+ = w_i \cdot \frac{1}{rows_i} \cdot LP$
end for
if $(TP > 0)$ return 0
else return 1

[1]For applying Markov Logic Networks as discussed in Section 3, MLN weights can be obtained by multiplying the logistic regression weights by the log-conditional probabilities.

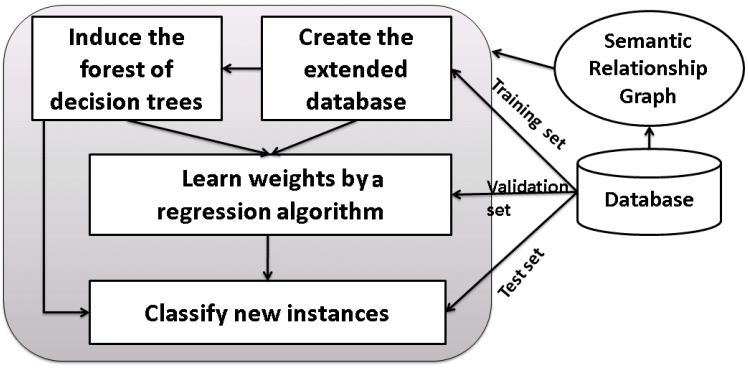

Figure 6: The process of classifying relational data using the Decision Forest

6. Learning Decision Forests With Regression

The log-linear classifier of Algorithm 1 requires as input three components. We discuss the construction of each of these components in turn.

(1) For constructing the extended database, in our experimental datasets it suffices simply to enumerate the possible valid joins based on Semantic Relationship Graphs and add the corresponding join tables as views to the database. The foreign key constraints keep the space of valid table joins manageable.

(2) For the classifier $\mathcal{C}_i$ for the join table J_i, we used probability estimation trees with the Laplace correction and no post-pruning, as recommended by Provost and Domingos [6].

(3) To learn the regression weights for combining the contributions from different table classifiers, we apply logistic regression as follows. Define b_i by

$$b_T = \log\left(\frac{P(c(t)=0|\boldsymbol{a}(t))}{P(c(t)=1|\boldsymbol{a}(t))}\right), b_i = \frac{1}{rows_i}\sum_{r=1}^{rows_i}\log\left(\frac{P(c(t)=0|J_{i,r})\cdot P(c(t)=1)}{P(c(t)=1|J_{i,r})\cdot P(c(t)=0)}\right).$$

Then Formula 5 can be rewritten as a logistic regression formula (Logit of posterior odds)

$$\log\left(\frac{P(c=0|T,J_1,\ldots,J_m}{P(c=1|T,J_1,\ldots,J_m)}\right) = w_0 + w_T b_T + w_1 b_1 + \ldots + w_m b_m. \qquad (6)$$

The regression model serves to combine the predictions of the different table classifiers. Whereas there are in general many options for combining

Algorithm 2 Multi-Relational Simple Decision Forest Induction and meta regression weight learning (implements Formula (5)

Input: Extended join tables $J_1, \ldots, J_m$.
Output:
(1) A forest of decision tree: probabilistic Classifier $\mathcal{C}_T$ for target table, $\mathcal{C}_i$ for each extended table.
(2) Regression weights $\vec{w}$.
Call: decision tree learner DT, and logistic regression weight learner LR

Fix a training set for DT learning, and a validation set for LR.
{Start Decision Trees induction} $\mathcal{C}_T :=$ Call DT (target table T).
for each table J_i in D **do**
  $\mathcal{C}_i :=$ Call DT (J_i).
end for
{Start Logistic Regression.} Create matrix M with $m+1$ columns.
for each target object t_k in the validation set with the class label $c(t_k)$ **do**
  $M_{k,0} := c(t_k)$
  $M_{k,1} := \log(\mathcal{C}_T(c(t_k) = 0|a(t_k)) - \log(\mathcal{C}(c(t_k) = 1|a(t_k))$
  for all join tables J_i **do**
    $LP := 0$
    for each row $J_{i,r}$ containing the target entity t_k **do**
      $LP+ = \log\left(\frac{\mathcal{C}_i(c(t_k)=0)|J_{i,r})\cdot\mathcal{C}_i(c(t_k)=1)}{\mathcal{C}_i(c(t_k)=1)|J_{i,r})\cdot\mathcal{C}_i(c(t_k)=0)}\right)$
    end for
    $M_{k,i} := \frac{1}{rows_i} \cdot LP$
  end for
end for
$\vec{w} =$ Call LR(M)

classifier predictions [28],[27, Ch.14], *the features of the log-linear model (the b_i values) follow from the Inter-Table Independence Assumption of Definition 1.*

Algorithm 2 describes the model learning phase. First, we divide the learning input data into training and validation set using a division of the original target table. We learn the classifiers on the training set and the weights on the validation set, to avoid overfitting. In lines 1 to 5 different trees on each table in the extended database are learned. To learn the regression weights, for each instance t_k in the set, feature vectors or independent variables $b_1(t_k), \ldots, b_m(t_k)$ are computed. A matrix with one row for each training instance t_k, one column for the class label $c(t_k)$, and one column for each predictor $b_i(t_k)$ is formed. This matrix is the input for a logistic regression package. Figure 6 represents the whole process of classifying relational data using the decision forest.

The run-time complexity of the method is dominated by the cost of applying a decision tree learner to different join tables, which in turn depends essentially on the size of the table joins. Experience with join-based methods [4, 5, 1] indicates that the table join sizes are manageable, for the following reasons: (i) Informative correlations seldom require more than a foreign key path of length 3, which correspond to joins of 3 tables or less. (ii) The tuple ID propagation technique is an efficient virtual join method that finds the sufficient statistics for learning without materializing the actual table join [1].

7. Evaluation

In this section, we compare different configurations of our proposed model with various relational classification models. We describe the datasets, basic setting of our experiments, and results in different evaluation metrics. Our code and datasets are available for anonymous ftp download from ftp://ftp.fas.sfu.ca/pub/cs/oschulte/sdf. The hypotheses we investigate are as follows.

1. That assuming (conditional) independence between only join tables leads to better use of the database information than assuming independence between attributes (as in the multi-relational Naive Bayes classifier) or randomly selecting attributes.

2. That the induction or learning time of methods making independence assumptions should be much faster than more general methods that do not.
3. That logistic regression applied to the log-contributions of linked tables is an effective way of pruning uninformative tables.

7.1. Datasets

We use five benchmark real-world datasets. The datasets feature both many-to-many and self-join relationships, as indicated, as well as 2-class and 3-class problems. The semantic relationship graphs for the datasets are depicted in Figure 7.

Financial Dataset. This dataset was used in the PKDD CUP 1999. *Loan* is the target table with 682 instances (606 of loans are successful). Since 86% of the examples are positive, the data distribution is quite skewed. We followed the modifications of Yin and Han [18, 4] and randomly selected 324 positive instances, and all the negative loans to make the numbers of positive tuples and negative tuples more balanced. The number of join tables in the extended database was 8.

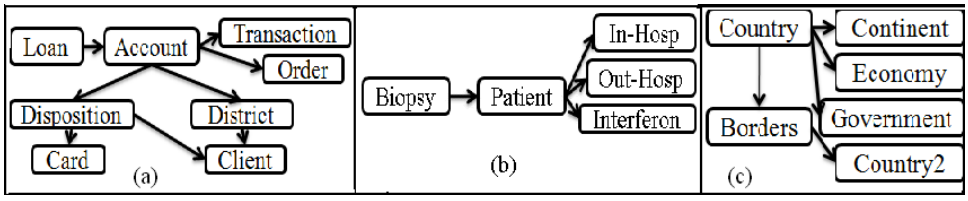

Figure 7: Semantic Relationship Graphs: (a)Financial, (b)Hepatitis, and (c)Mondial dataset.

Hepatitis Database. This data is a modified version of the PKDD'02 Discovery Challenge database. We followed the modification of Frank *et al.*[39]. *Biopsy* is the target table with 206 instances of Hepatitis B, and 484 cases of Hepatitis C. The number of join tables in the extended database was 4.

Mondial Database. This dataset contains data from multiple geographical web data sources [40]. We predict the religion of a country as Christian (positive) with 114 instances vs. all other religions with 71 instances. We followed the modification of She *et al.* [41]. We use a subset of the tables and features. *Borders* is a many-many relationship between Country and

Country. To create an acyclic semantic relationship graph for this database, we duplicated the Country table (cf. [18, 4]). The number of join tables in the extended database was 5.

MovieLens. This dataset is drawn from the UC Irvine machine learning repository. It contains two entity tables: *User* with 941 tuples and *Item*, with 1,682 tuples, and one relationship table *Rated* with 80,000 ratings. The table *Item* has 17 Boolean attributes that indicate the genres of a given movie. The class label is the user attribute *age* that we discretized into three bins with equal frequency.

JMDB. This is our most complex dataset, containing information from the Internet Movie Database (IMDB), about movies, such as titles, ratings, actors, studios. We obtained it from the IMDB data interface (http://www.imdb.com/interfaces). The target table is *ratings*, and the target attribute *rank* records users, on average, rank a movie on a scale from 1 to 10. We discretized the ratings into 3 equal-width intervals, and removed movies that had not been ranked by any user, for a total of 281,449 target instances. We omitted textual information, so our orginal database contained 7 tables, with an additional 6 join tables in the extended database.

7.2. Experimental Setting, Systems, and Performance Metrics.

All experiments were done on a Pentium 4 CPU 2.8Ghz and 3GB of RAM system (except for some on the JMDB dataset, see below). The implementation used many of the procedures of the data mining software Weka [42]. We compared the following relational classifiers. The first three are variations of our own framework and the last three classifiers were developed by other researchers.

Normalized decision forest Weighted log linear decision forest with scaling factors (Formula 5).

Unnormalized decision forest weighted log linear decision forest with no scaled weights (Formula 5 with $rows_i = 1$).

Naive decision forest Decision forest with no weights for the trees (Formula 4).

TILDE First-order decision tree [19].

FORF-NA First-order random forest with no aggregates [12].

Graph-NB Multi-relational naive Bayes classifier [4].

TreeLiker-Relf A propositionalization algorithm that constructs tree-like nonredundant conjunctive features [16]. We apply a decision tree learner to the constructed features.

TreeLiker-Poly An extension of TreeLiker that supports the use of aggregate functions for numeric attributes [16].

The datasets MovieLens and JMDB feature class labels with 3 possible values. We used a multinomial logistic regression model, which learns 2 log-odds weight vectors with class 3 as the pivot (log-odds of 1 vs. 3 and log-odds of 2 vs. 3). We converted the log-odds to probabilities using the standard formula for the multinomial logistic regression model, and classified instances according to the resulting probability ranking of classes. We made use of the following implementations.

Decision Tree Learning Weka's J48 as a decision tree learner on each join table. For simple decision forests, we used the probability estimation tree setting that turns off pruning and applies the Laplace correction [6]. We applied the classification tree setting with the features generated by TreeLiker.

Logistic Regression The simple logistic regression procedure of Weka.

Graph-NB We implemented Graph-NB using the single-table Naive Bayes Classifier procedure of Weka as the code is not available.

TreeLiker We use the code provided by the system creators, available at http://ida.felk.cvut.cz/treeliker/. The minimum frequency parameter is set at the default value 1%.

This design evaluates two different ways of upgrading the same propositional decision tree learner to relational data: using the cross-table Naive Bayes assumption, as in simple decision forests, vs. using propositionalized features, as in Treeliker.

TILDE and FORF-NA are included in the ACE data mining system [43]. We ran TILDE with the default setting. For FORF-NA we used bagging,

chose 25% of features to consider for testing in each node, and fixed the number of trees in the forest to 33. Vens *et al.* report that this setting leads to the most efficient results in terms of accuracy and running time [12]. As in previous studies [19, 12, 4], we use a leave-one-out test, where the prediction for each target entity is based on knowledge of all other entities, and we average the predictions over all target entities in the test fold. For decision forests and Graph-NB, we performed ten-fold cross-validation to evaluate predictive accuracy. In each run we learn the decision tree on a random 8-fold of data, learn the weights of logistic regression on a 1 fold (validation set), and test the model on the remaining fold. For the other systems, we used the evaluation procedures supplied with the systems as recommended by their creators. (Cross-validation for TILDE , out-of-bag for FORF-NA.)

To evaluate classification performance, we used the following metrics:

Run time Induction time of the model.

Accuracy Percentage of correctly classified instances.

AUC The area under the ROC curve.

Weighted F-measure Sum of the F-measures for each class label, each weighted according to the number of instances with a particular class label (class prior). F-measure is the weighted harmonic mean of precision and recall.

7.3. Results.

We discuss run times for learning, then predictive performance.

7.3.1. Learning Times.

Table 1 reports the induction times of different relational classifiers on the three datasets. The fastest system is Graph-NB, the multi-relational Naive Bayes classifier, which makes the strongest independence assumptions. For Normalized and Unnormalized Decision Forests, the induction time is basically the sum of the runtimes of the Naive Decision Forest learner and the logistic regression. Normalized and Unnormalized Decision Forest differ only in using scaling factors so they have the same induction time. The Decision Forest learners are very fast as well, but because the Simple Decision Forest learner considers dependencies within tables, they have a slightly longer induction time compared to the Multi-relational Naive Bayes classifier. The

fast learning times of the independence-based methods on the large JMDB dataset illustrate that these methods scale well in the dataset size.

The systems based on independence assumptions are 1,000 times or more faster than the older propositionalization-based systems TILDE and FORF-NA, which is a dramatic improvement. The state-of-the-art TreeLiker method is quite fast on the smaller datasets, still about an order of magnitude slower than decision forest learning. Table 7.3.1 shows the number of relevant features constructed by the Treeliker methods. Because TreeLiker constructs a large set of features, which correspond to subsets (conjunctions) of attributes, it does not scale well with the size of the dataset, which is illustrated by its learning time on the JMDB database. TreeLiker did not terminate on JMDB using our standard system, so to obtain the classification results shown, we ran it on a high-performance cluster with 554 nodes. In sum, our simulations provide evidence that the cross-table independence assumption of Definition 1 allows learning to proceed efficiently by analyzing separate join tables independently and combining the results in a principled manner. In contrast, propositionalization methods search a large feature space, and even a state-of-the-art efficient method like TreeLiker does not scale well with dataset size.

Learning Time	Simple Decision Forest			Reference Methods				
Dataset	Normalized	Unnormalized	Naive	TILDE	FORF-NA	Graph-NB	TreeLiker-Relf	TreeLiker-Poly
Financial	2.3	2.3	1.4	2429	54006	**0.8**	61.5	41.3
Hepatitis	1.1	1.1	0.54	853	10515	**0.21**	7.24	6.43
Mondial	0.26	0.26	0.25	0.3	7.07	**0.18**	11.97	9.51
MovieLens	2.2	2.2	2	3	20	**1.3**	17.51	16.5
JMDB	13	13	11	NT	NT	**9**	3173.1	506.67

Table 1: Average Induction time of different algorithms in seconds. Normalized and Unnormalized Decision Forests use logistic regression to learn weights for join tables. NT denotes nontermination after 4 days of running. The TreeLiker results on JMDB were obtained on a high-performance cluster, as described in the text.

No.of Attributes	TreeLiker - Relf	Treeliker - Poly
MovieLens	53	26
Financial	337	353
Hepatitis	305	185
Mondial	418	40
JMDB	44	33

Table 2: Number of Attributes constructed by the TreeLiker propositionalization methods

7.3.2. Predictive Performance.

Table 3 shows the Accuracy, AUC, and F-measure of the different classifiers on the three datasets. For the multi-class problems, we report only accuracy, since there is no standard way to extend f-measure and AUC to multi-class problems [44], and since the three measures are highly correlated on the binary class problems. We make the following observations.

1. Overall, the Normalized Decision Forest achieves always good classification performance and typically the best.
2. Comparing Naive Decision Forest with Naive Bayes net classifier (Graph-NB), taking into account *within-table* dependencies between attributes of each join table is clearly beneficial. Even the worst Decision Forest method makes better predictions.
3. Comparing Decision Forests with Random Forests, we found that grouping together the features of each join table for a classifier, instead of learning on randomly selected subset of features, improved the performance substantially on two of the three datasets (Hepatitis and Mondial). On Financial, the Normalized and Naive Decision Forest methods achieved better performance.

Regression Weights. To show how linear regression assigns weights to the information from different tables, the weights learned by Normalized and Unnormalized decision forests for the target table and each extended table of each dataset are listed in Table 4. There are two extended join tables involving the Client relation, since in the Semantic Relationship Graph of Financial there are two pathways to the Client table (see Figure 7). Because both join tables receive regression weights 0, we show only one as "Client". The fact that the nonzero weights are far from uniform shows that regression learns an importance ranking of the information from different tables. The 0 weights demonstrate how regression prunes uninformative join tables. For instance, on the Mondial database, the weights indicate that the most important factor in predicting the majority religion of a country is the majority religion of its neighbors (1.43), the second most important factor is the continent on which the country is located (1.23), and the third are the attributes of the country contained in the country target table (0.94). If knowledge discovery is the goal of data analysis, the regression weights enhance the information conveyed by the decision trees, or extracted rules (see Section 3.2).

Financial	Simple Decision Forest			Reference Methods				
Method	Normalized	Unnormalized	Naive	TILDE	FORF-NA	Graph-NB	TreeLiker-Relf	TreeLiker-Poly
Accuracy	**92%**	87%	91%	89%	89%	81%	87%	88%
AUC	**0.88**	0.85	0.85	0.69	0.75	0.82	0.67	0.58
F-measure	**0.89**	0.84	**0.89**	0.88	0.87	0.79	0.85	0.86

Hepatitis	Simple Decision Forest			Reference Methods				
Method	Normalized	Unnormalized	Naive	TILDE	FORF-NA	Graph-NB	TreeLiker-Relf	TreeLiker-Poly
Accuracy	**84%**	**84%**	80%	61%	63%	75%	69%	67%
AUC	**0.88**	0.86	0.80	0.61	0.64	0.79	0.74	0.69
F-measure	**0.79**	0.75	0.75	0.59	0.61	0.68	0.69	0.67

Mondial	Simple Decision Forest			Reference Methods				
Method	Normalized	Unnormalized	Naive	TILDE	FORF-NA	Graph-NB	TreeLiker-Relf	TreeLiker-Poly
Accuracy	**84%**	**84%**	83%	71%	71%	73%	80%	77%
AUC	0.85	**0.86**	**0.86**	0.75	0.79	0.74	0.833	0.702
F-measure	**0.83**	0.82	0.81	0.78	0.77	0.75	0.79	0.74

Multi-Class	Simple Decision Forest			Reference Methods				
Method	Normalized	Unnormalized	Naive	TILDE	FORF-NA	Graph-NB	TreeLiker-Relf	TreeLiker-Poly
MovieLens-Acc	61.5%	60%	62%	47%	47%	51%	61%	**63%**
JMDB-Acc	**57%**	56%	53%	NT	NT	51.7%	51%	51%

Table 3: Performance of different classifiers by dataset. For multi-class problems we report accuracy only.

8. Conclusion

A goal of relational classification is to make predictions that utilize information not only about the target table but also about related objects. Decisions trees are a well-established predictive method for propositional single table data. We proposed a new way of upgrading them for relational data classification. The basic idea is to independently learn different decision trees for different related tables, and then combine their contributions in a new log-linear model to predict class probabilities. The log-linear model is derived from an explicitly defined cross-table Naive Bayes independence assumption. Features that distinguish this method from other relational decision tree learners include the following. (1) Aggregation functions are not used, which avoids some information loss and allows for efficient learning. (2) Information from all links is considered in classification. (3) Logistic regression is used to weight information from different tables. Empirical evaluation on three datasets showed very fast runtimes, with improved predictive performance.

A natural variant of our approach, especially for continuous attributes,

Financial	w_0	Loan	Account	Order	Trans	Disp	District	Card	Client
Normalized DF	-0.19	0.52	0	0	0.2	0.02	0	0	0
Unnormalized DF	-0.34	0.38	0	0.34	0.2	0	0	0	0

Hepatitis	w_0	Biopsy	Patient	In-Hosp	Out-Hosp	Interferon
Normalized DF	0.29	0.3	0.2	0.9	0	0.3
Unnormalized DF	0.97	0.76	0.4	0.22	0.03	0.9

Mondial	w_0	Country	Borders	Country2	Continent	Economy	Government
Normalized DF	0	0.94	0	1.43	1.23	0.86	0.7
Unnormalized DF	0	0.8	0.3	1.09	1.1	0.79	0

Table 4: The regression weights indicating the importance of linked tables for datasets with binary class labels. Normalized DF divides a weights by the size of the associated join table.

is to use logistic regression as the base probabilistic classifier, instead of decision trees. This would provide regression weights on attributes/features within each table in addition to the weights for each table we learned in our experiment.

A promising direction for future work is to combine our log-linear model with propositionalization techniques. While a search for informative aggregate features is computationally expensive, when it succeeds, the new aggregate features can increase the predictive accuracy (e.g., [39, 12]). There are several possibilities for a combined hybrid approach. (i) Once good aggregate features are found, they can be treated like other features and used in a decision tree. (ii) A simple decision forest is fast to learn and can establish a strong baseline for evaluating the information gain due to a candidate aggregate feature. (iii) The regression weights can be used to quickly prune uninformative join tables with 0 or small weights, which allows the search for aggregate features to focus on the most relevant link paths.

Acknowledgements

This research was supported by a Discovery grant to the senior author by the Natural Sciences and Engineering Research Council of Canada. Anonymous reviewers for *Decision Support Systems* provided helpful comments. We are grateful to Ondvrej Kuvzelka for help with running Treeliker.

References

[1] X. Yin, J. Han, Exploring the power of heuristics and links in multi-relational data mining, in: ISMIS, LNAI, Springer, 2008, pp. 17–27.

[2] J. Neville, D. Jensen, Relational dependency networks, in: Introduction to Statistical Relational Learning [45], Ch. 8, pp. 239–268.

[3] J. Neville, D. Jensen, B. Gallagher, R. Fairgrieve, Simple estimators for relational bayesian classifiers, in: ICDM, IEEE Computer Society, 2003, pp. 609–612.

[4] H. Chen, H. Liu, J. Han, X. Yin, Exploring optimization of semantic relationship graph for multi-relational Bayesian classification, Decision Support Systems 48 (1) (2009) 112–121.

[5] G. Manjunath, M. N. Murty, D. Sitaram, A practical heterogeneous classifier for relational databases, in: ICPR, IEEE Computer Society, 2010, pp. 3316–3319.

[6] F. J. Provost, P. Domingos, Tree induction for probability-based ranking, Machine Learning 52 (3) (2003) 199–215.

[7] D. Fierens, J. Ramon, H. Blockeel, M. Bruynooghe, A comparison of pruning criteria for probability trees, Machine Learning 78 (1-2) (2010) 251–285.

[8] H. Zhang, J. Su, Conditional independence trees, in: ECML, LNAI, Springer, 2004, pp. 513–524.

[9] R. Kohavi, Scaling up the accuracy of naive-bayes classifiers: A decision-tree hybrid, in: KDD, AAAI Press, 1996, pp. 202–207.

[10] P. Domingos, D. Lowd, Markov Logic: An Interface Layer for Artificial Intelligence, Morgan and Claypool Publishers, 2009.

[11] M. Brydon, A. Gemino, You've data mined. now what?, The Communications of the Association for Information Systems 22 (33) (2008) 603–616.

[12] A. Van Assche, C. Vens, H. Blockeel, S. Dvzeroski, First order random forests: Learning relational classifiers with complex aggregates, Machine Learning 64 (1) (2006) 149–182.

[13] S. Kramer, N. Lavrac, P. Flach, Propositionalization approaches to relational data mining, in: Relational Data Mining, Springer, 2000, pp. 262–286.

[14] J. Neville, D. Jensen, L. Friedland, M. Hay, Learning relational probability trees, in: KDD, ACM Press, 2003, pp. 625–630.

[15] A. Popescul, L. Ungar, Feature generation and selection in multi-relational learning, in: Introduction to Statistical Relational Learning [45], Ch. 16, pp. 453–476.

[16] O. Kuzelka, F. Zelezný, Block-wise construction of tree-like relational features with monotone reducibility and redundancy, Machine Learning 83 (2) (2011) 163–192.

[17] N. Landwehr, K. Kersting, L. D. Raedt, nfoil: Integrating naïve bayes and foil, in: AAAI, AAAI Press, 2005, pp. 795–800.

[18] X. Yin, J. Han, J. Yang, P. S. Yu, Crossmine: Efficient classification across multiple database relations, in: ICDE, IEEE Computer Society, 2004, pp. 399–410.

[19] H. Blockeel, L. D. Raedt, Top-down induction of first-order logical decision trees, Artificial Intelligence 101 (1-2) (1998) 285–297.

[20] J. R. Quinlan, C4.5: programs for machine learning, Morgan Kaufmann Publishers Inc., 1993.

[21] A. Atramentov, H. Leiva, V. Honavar, A multi-relational decision tree learning algorithm - implementation and experiments, in: ILP, Vol. 2835 of LNAI, Springer, 2003, pp. 38–56.

[22] J.-F. Guo, J. Li, W.-F. Bian, An efficient relational decision tree classification algorithm, in: ICNC, Vol. 3, IEEE Computer Society, 2007, pp. 530–534.

[23] A. Y. Ng, M. I. Jordan, On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes, in: NIPS, Vol. 14, MIT Press, 2001, pp. 841–848.

[24] Q. Lu, L. Getoor, Link-based classification, in: ICML, AAAI Press, 2003, pp. 496–503.

[25] M. Ceci, A. Appice, D. Malerba, Mr-SBC: A multi-relational naïve Bayes classifier, in: PKDD, Vol. 2838 of LNAI, Springer, 2003, pp. 95–106.

[26] P. A. Flach, N. Lachiche, Naive Bayesian classification of structured data, Machine Learning. 57 (3) (2004) 233–269.

[27] C. M. Bishop, Pattern Recognition and Machine Learning, Springer, 2006.

[28] L. I. Kuncheva, Combining Pattern Classifiers: Methods and Algorithms, Wiley, 2004.

[29] L. Getoor, N. Friedman, D. Koller, A. Pfeffer, B. Taskar, Probabilistic relational models, in: Introduction to Statistical Relational Learning [45], Ch. 5, pp. 129–173.

[30] B. Taskar, P. Abbeel, D. Koller, Discriminative probabilistic models for relational data, in: UAI, Morgan Kaufmann Publishers Inc., 2002, pp. 485–492.

[31] T. N. Huynh, R. J. Mooney, Discriminative structure and parameter learning for markov logic networks, in: ICML, Vol. 307, ACM, 2008, pp. 416–423.

[32] O. Schulte, H. Khosravi, Learning graphical models for relational data via lattice search, Machine Learning 88 (3) (2012) 331–368.

[33] T. Khot, S. Natarajan, K. Kersting, J. W. Shavlik, Learning markov logic networks via functional gradient boosting, in: ICDM, IEEE Computer Society, 2011, pp. 320–329.

[34] D. Jensen, J. Neville, B. Gallagher, Why collective inference improves relational classification, in: SIGKDD, ACM Press, 2004, pp. 593–598.

[35] J. D. Ullman, Principles of database systems, 2nd Edition, W. H. Freeman & Co., 1982.

[36] H. Khosravi, O. Schulte, J. Hu, T. Gao, Learning compact markov logic networks with decision trees, Machine Learning 89 (3) (2012) 257–277.

[37] P. Domingos, M. Pazzani, Beyond independence: Conditions for the optimality of the simple Bayesian classifier, in: ICML, Morgan Kaufmann, 1996, pp. 105–112.

[38] R. Raina, Y. Shen, A. Y. Ng, A. Mccallum, Classification with hybrid generative/discriminative models, in: NIPS, MIT Press, 2003, pp. 545–552.

[39] R. Frank, F. Moser, M. Ester, A method for multi-relational classification using single and multi-feature aggregation functions, in: PKDD, Vol. 4702 of LNAI, Springer, 2007, pp. 430–437.

[40] W. May, Information extraction and integration: The mondial case study, Tech. rep., Universität Freiburg, Institut für Informatik (1999).

[41] R. She, K. Wang, Y. Xu, P. S. Yu, Pushing feature selection ahead of join, in: SIAM SDM, 2005, pp. 536–540.

[42] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I. H. Witten, The weka data mining software: an update, SIGKDD Explorations 11 (1) (2009) 10–18.

[43] H. Blockeel, L. Dehaspe, J. Ramon, J. Struyf, A. Van Assche, C. Vens, D. Fierens, The ACE Data Mining System: User's Manual, http://dtai.cs.kuleuven.be/ACE/doc/ACEuser-1.2.16.pdf (2009).

[44] R. Espíndola, N. Ebecken, On extending f-measure and g-mean metrics to multi-class problems, Data mining VI: Data mining, text mining and their business applications 35 (2005) 25–34.

[45] L. Getoor, B. Taskar, Introduction to statistical relational learning, MIT Press, 2007.