A Heterogeneous Naive-Bayesian Classifier for Relational Databases

Geetha Manjunath, M Narasimha Murty, Dinkar Sitaram

HP Laboratories
HPL-2009-225

Keyword(s):
Relational databases, Classification, Data Mining, RDF

Abstract:

Most enterprise data is distributed in multiple relational databases with expert-designed schema. Application of single-table data mining techniques to distributed relational data not only incurs a computational penalty for converting to a "at" form (mega-join), even the human-specified semantic information present in the relations/schema is lost. Purely relational classification algorithms on the other hand, do consider detailed relationships between attributes. However, these techniques either require computationally intensive transformations or multiple analysis of fused datasets, which becomes infeasible in practical scenarios. Classification being one of the most popular predictive data mining tasks, we need practical algorithms that can be directly applied on existing databases. We present such a practical two-phase classification algorithm for relational databases with a semantic divide and conquer approach. We propose and prove a recursive, prediction aggregation technique over heterogeneous classifiers applied on individual tables. Our approach also attempts to effectively leverage the semantic knowledge of the application that is hidden in the database schema using the Join Graph of an application. To automate the classification process, RDF (the core Semantic Web data model) is used for problem specification. A preliminary evaluation over TPCH and UCI benchmarks shows reduced training time in automated practical scenarios, without any loss of prediction accuracy. In fact, we show improved accuracy due to application of heterogeneous classifiers on individual tables by comparing it to other state-of-art techniques.

External Posting Date: September 6, 2009 [Fulltext] Approved for External Publication
Internal Posting Date: September 6, 2009 [Fulltext]

A Heterogeneous Naive-Bayesian Classifier for Relational Databases

Geetha Manjunath*
Hewlett Packard Labs
Bangalore, India
geetha.manjunath@hp.com

M Narasimha Murty
Indian Institute of Science
Bangalore, India
mnm@csa.iisc.ernet.in

Dinkar Sitaram
Hewlett Packard India
Bangalore, India
dinakar.sitaram@hp.com

ABSTRACT

Most enterprise data is distributed in multiple relational databases with expert-designed schema. Application of single-table data mining techniques to distributed relational data not only incurs a computational penalty for converting to a "flat" form (mega-join), even the human-specified semantic information present in the relations/schema is lost. Purely relational classification algorithms on the other hand, do consider detailed relationships between attributes. However, these techniques either require computationally intensive transformations or multiple analysis of fused datasets, which becomes infeasible in practical scenarios. Classification being one of the most popular predictive data mining tasks, we need practical algorithms that can be directly applied on existing databases.

We present such a practical two-phase classification algorithm for relational databases with a semantic divide and conquer approach. We propose and prove a recursive, prediction aggregation technique over heterogeneous classifiers applied on individual tables. Our approach also attempts to effectively leverage the semantic knowledge of the application that is hidden in the database schema using the Join Graph of an application. To automate the classification process, RDF (the core Semantic Web data model) is used for problem specification. A preliminary evaluation over TPCH and UCI benchmarks shows reduced training time in automated practical scenarios, without any loss of prediction accuracy. In fact, we show improved accuracy due to application of heterogeneous classifiers on individual tables by comparing it to other state-of-art techniques.

Categories and Subject Descriptors

I.5.2 [**Pattern Recognition**]: Design Methodology; H.2.8 [**Database Application**]: Miscellaneous

*This work is based on her work done at IISc.

Keywords

Relational databases, Classification, Data Mining, RDF

1. INTRODUCTION

Most data in real enterprises exists in multiple relational databases each with well-designed schema. Use of separate database tables to group related data attributes with foreign-keys capturing semantic relationship between the tables is a recommended database design approach. Proven methods of database normalization [23] to minimize redundancy due to repeated attribute values while optimizing query response times are commonly used. While this is the scene of real enterprise application data, most classical classification algorithms assume that all the attributes are available in a single table, a ŞflatT̃ data representation.

Application of single-table data mining techniques to distributed relational data not only incurs a computational penalty for converting to a "flat" form (mega-join), even the human-specified semantic information present in the relations/schema is lost. Relational classification algorithms on the other hand, do consider detailed relationships between attributes. However, use of inductive logic programming [18,19,20] or probabilistic relational models [21,26] based techniques, make these algorithms computationally expensive for complex databases as they need multiple database scans for repeated selection of best predicates and corresponding cross validation [20]. Further, many of these techniques cannot be directly applied on the relational tables, as they require propositionalization. We therefore need accurate classification algorithms that can work directly on relational data distributed across multiple database tables with well-designed schema.

A simple classifier based on Naive Bayes condition is known [22] to perform well on real single table data. However, when applied in a multi-relational setting, its accuracy suffers from the altered statistical information of individual attributes (priors for example) during the mega-join[1]. Of course, the basic assumption in these techniques is that all attributes in the relational tables are independent, which is rarely the case in real databases. One of the key benefits of designing multiple relational tables for an application (using E-R diagrams) is grouping of related attributes. A good classifier should therefore leverage this inherent relationship among attributes or the hidden semantics in the schema design.

We propose a novel, two-level, classification algorithm for relational databases that utilizes this hidden application knowledge from the database design. Using our approach, any classification algorithm can be employed on individual ta-

bles in the first level and these individual posteriors are combined at the second level to get the aggregated predictions. We propose and prove a recursive prediction-aggregation technique for combining individual posteriors at this second level. We believe that our scalable, heterogeneous classification algorithm is more suitable for real life applications since it directly works on existing databases and also assumes inter-table attribute independence. In fact, our approach supports use of different classifiers for different individual tables as well. With this, we show an order of magnitude improvement in training time on relational databases (TPCH benchmark) without reducing the classification accuracy (UCI benchmarks).. For the implementation of this algorithm, we use the core Semantic Web data model, RDF (Resource Description Framework) to enable semi-automatic application of the classification algorithm on existing relational databases.

The structure of this document is as follows. Section 2 details earlier work on classification of relational databases and related techniques. Section 3 states the pragmatic problem being solved and section 4 describes our proposed solution. We provide an induction-based proof for our modified Naive Bayes approach in section 5. Section 6 details the implementation and results we have on TPCH and UCI benchmarks. We conclude in section 7.

2. RELATED WORK

Peter Flach and Nicholas Lachiche in [5] provide a simple tutorial style introduction to Naive Bayesian classification of structured data. They describe three representations of structured data: (a) datalog style with data described using individuals, properties and structural predicates, (b) term-based representation with tuples and data functors and (c) relational representation with E-R diagrams. They propose two algorithms that extend Naive Bayes to handle the first two types of representations in the language space and individual space. Our algorithm handles the second case where structured data is represented using E-R diagrams for instance.

Cross-Mine [2] works directly on relational databases. It uses tuple-id propagation to virtually associate tuple-ids of the target relation to the tuple-ids of the non-target relation. They use a meta-learner over classifiers on individual tables which is very time consuming. We later compare our algorithm with theirs to show better performance of our heterogeneous classifier. Statistical relational learning models, specifically probabilistic relational models, have also been used in relational setting by extending Bayesian networks, but again require lot of operations on the fused dataset, which we believe is not practical. Other mechanisms for meta learning based on boosting [16], bagging [15] and stacking [17] require construction of different hypotheses from subsets of learning instances which do not scale on practical databases. Use of Naive Bayes assumption across transformational views of relational databases in our solution, removes the need for any computation on the fused dataset.

Current approaches that use Naive Bayes approach [4,5] employ a two step algorithm, wherein the first step extracts the relevant first-order features or rules using ILP and the second step computes the final probability using Naive-Bayes rule using extracted features. Graph-NB [1] is another modified Naive Bayes algorithm for classifying relational datasets that works directly on databases. However, their approach assumes mutual independence of all attributes in the dataset and also requires the probability of table linkages, which makes it somewhat impractical to apply on real databases. Further, they do not have any proofs of correctness or even a practical algorithm or results in that direction. They use a Semantic Relationship Graph to represent relationships between tables and propose an algorithm to prune this graph if found complex. This graph is at a broad level similar to our concept of a Join Graph that specifies the knowledge about an application. However, they use the graph only for feature elimination.

There have been many practical approaches to associative rule mining over relational database systems. A technique of associative classification [29,30] where association rules are generated over databases and analyzed for classification, have been proposed. However such techniques typically work well for datasets with nominal attributes and also require analysis on fused dataset.

Multi-view learning techniques have been applied to solve information extraction [12] and text classification [13] problems by creating views to represent set of disjoint feature sets of a specific entity. Hongyu Guo, et al [3] propose use of multi-view methods and formulate concept learning in multi-relational databases. While, use of SQL views to capture the dataset for first level classification is somewhat similar to ours, they use additional aggregation based features (COUNT, SUM, AVG) to summarize properties of a set of related objects. Our solution based on the proof provided in section 5 requires only prior probabilities (just COUNT) on the fused dataset.

More complex algorithms to efficiently classify relational data have been tried as well. R Alfred, et al [10] use a genetic semi-supervised clustering algorithm to aggregate data in multiple tables. In [24], Neville and Jensen describe an iterative classification procedure that exploits specified relationships in relational data for getting greater classification accuracy with a simple Bayesian classifier that dynamically updates the attributes of some objects as high confidence inferences are made about related objects. Taskar, et al, present a new approach [11] of using a single probabilistic model for the entire database that captures interactions between instances in the domain.

3. THE PRACTICAL PROBLEM

We would like to come up with an efficient and practical solution for classifying enterprise data in relational databases. First of all, the algorithm should be scalable in the size of the database tables as well as the number of such tables. It should work directly out of existing databases since it is not feasible (from both time and space perspective) to transform data even in individual tables to different forms of representation. It should be able to work without requiring collation or replication of data from all tables. Since there may be different authorization controls for different databases tables, the algorithm should be modular with an ability to execute different parts of the algorithm by different users/stages. Further, the algorithm should effectively leverage the semantic grouping that is implicit in the design of the RDBMS. Since, it is typical to have related attributes grouped within tables, no specific assumptions on the independence of the attributes or any specific probability distribution should be made for attributes of a single table.

Table 1: Researcher

ENo	sex	age	UNo	status
E1	F	38	U1	Y
E2	M	51	U2	N
E3	M	42	U3	Y
E4	F	35	U4	N
E5	M	22	U1	?

Table 2: Unit (with propagated label)

UNo	rank	size	
U1	2	>10	Y
U2	2	>10	N
U3	1	<10	Y
U4	2	<10	N

Table 3: Journal Papers (with propagated labels)

PNo	type	level	Author	
p1	conference	1	E1	Y
p2	conference	2	E2	N
p3	conference	3	E3	Y
p4	journal	1	E1	Y
p5	journal	1	E4	N
p6	journal	2	E2	N
p7	conference	2	E5	

Table 4: Fused Table

ENo	sex	age	PNo	type	lev	UNo	rank	Size	stat
E1	F	38	p1	Conf	1	U1	2	>10	Y
E2	M	51	p2	Conf	2	U2	2	>10	N
E3	M	42	p3	Conf	3	U3	1	<10	Y
E1	F	38	p4	Jour	1	U1	2	>10	Y
E4	F	35	p5	Jour	1	U4	2	<10	N
E2	F	51	p6	jour	2	U2	2	>10	N
E5	M	22	p7	conf	2	U1	2	>10	?

Figure 1: Example Relational Database

Finally, from a practical usage perspective, the implementation should require minimum manual intervention to apply the solution on existing databases.

Consider a sample application containing three database tables Researcher, Unit, and Journal shown in Figure 1. The classification under interest is that of predicting the readiness of the researcher for a promotion. Researcher E5 is an example test case whose status needs to be determined by the classification algorithm. One approach is to just fuse all the tables with a mega join and to create the fused Table 4. Now a conventional classification algorithm can be applied over this to determine the class label of the test case (row E5). However, as described earlier, this approach is not feasible in real enterprises due to scalability, time and security reasons. We therefore need to determine the class label of researcher E5 directly using multiple database tables which include relevant data for classification.

In a well designed database, this class label or the attribute of interest (called target attribute) will be just in one of the tables of the database (here table 1). The dotted column in tables 2 and 3 shows the propagated class label during data preparation (described in section 4.1). The classification problem now is to use these individual tables with propagated class labels as the training set and derive the class labels for the test dataset (basically fill the records with missing attribute value).

4. PROPOSED SOLUTION

We propose a two-phase classification algorithm for relational databases with a semantic divide and conquer approach. We use a new way of combining probabilistic predictions from first level classification on individual tables. The aggregation mechanism is recursive over a graph showing table dependencies (formally defined later) and is inductively derived from Naive Bayes rule as detailed in the proof (section 5). We believe that the proposed approach is more realistic for typical enterprise applications than pure Naive Bayesian approach as it assumes attribute independence only across tables, and not among attributes within individual tables. Further, any stochastic classification algorithm can be independently employed on individual tables (based on the nature of the attributes within each table) in our approach, enabling modular heterogeneous classification.

Expert database designers typically group sets of related attributes of an entity into separate database tables. Attributes in different tables are usually independent and when not so, the relationships between the tables are explicitly specified with the use of foreign keys. While this division is mainly used to reduce duplicate entries (schema normalization) and improve query performance, the relational schema does give an important hint on the semantics of the database application. We use this implicit knowledge of attribute dependencies to generate appropriate views of the database tables for table-level classification in the first phase of our algorithm.

In the second level of classification, we compute the aggregated likelihood for a particular class by collating the class priors and posteriors obtained from the different table-level classifiers. The collated likelihood ratio at every table is just based on its own records and records from its immediately related tables (those that can be joined with it directly). Operations on collated data is completely avoided.

We present further details of the above solution in the rest of this section.

4.1 Preparing the Data

As mentioned earlier, data used by a typical enterprise application consists of multiple database tables and in a well designed database, the class label or the attribute of interest (called target attribute) will be in one of these tables (called the target table). So, the first task of data preparation is to automatically propagate this attribute to the rest of the tables. In the sample application, the attribute *status* is the target attribute and table Researcher is the target table. The columns containing the propagated attribute for tables 2 and 3 are shown with dotted border in Figure 1.

Since database design hugely impacts the enterprise application performance, lot of care is taken at the design of the database schema itself to group semantically related attributes into one table and explicitly specifying the relationships across tables (through E-R diagrams). We exploit this expert knowledge to enhance classification accuracy as well as its speed. Each table has a primary key, a unique attribute within the table that can be used as a record iden-

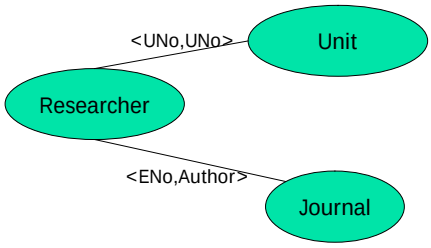

Figure 2: The Join Graph for the example

tifier. If a table needs to refer to a related attribute, it uses the primary key of another table as an additional column or attribute, called as a foreign key. We use the concept of a Join Graph to represent this semantic dependency between database tables.

DEFINITION: A Join Graph is an undirected, labeled graph (V, E, W), where V is a set of vertices, one vertex per table in the database, E is a set of labeled edges. An edge (V_i, V_j) *with label* $< l_i, l_j > \in W$ *exists if Table* V_i *can be directly joined with Table* V_j *using a JOIN clause* $T_i.l_i = T_j.l_j$ *. W is a set of attribute label pairs, each of which links two tables through one of the following*

- *Primary-key to foreign-key relationship*
- *Foreign-key to primary-key relationship*
- *Expert-specified explicit attribute relationship*

The Join Graph corresponding to the example tables in Figure 1 is shown in Figure 2. Join Graph will be the primary source of information about the semantics of the enterprise application. It will be used in our solution for both automatic class label propagation as well as directing the different phases of our classification algorithm.

We use standard database views in order to propagate the class labels from target table to the other tables. The pre-processing phase automatically generates the CREATE VIEW statements using the information provided in the Join Graph. For example, we create an extended Unit table using

```
CREATE VIEW Unit_ext FROM Unit, Researcher
SELECT UNo, rank, size, status
WHERE Researcher.UNo = Unit.UNo
```

As may be obvious from Figure 1, the extended table consists of all the relevant attributes of the original table and an additional new attribute with propagated values of the target attribute. An obvious advantage of using database views to do the class label propagation is the liveness of the transformation. If the original databases are updated, the views also get correspondingly updated by definition. Even 1-to-many and many-to-1 relationships are automatically handled with appropriately duplicated rows in the views. This makes it more convenient for practical scenarios. Additionally, if there are any authorization restrictions on some databases, the same authorization will be extended to the views as well, and the first-level classification on this live view can be performed by the authorized user.

Real databases may have more complicated Join Graphs and require a simple breadth-first traversal of the Join Graph (starting from the target table) to remove edges causing cycles. A topological sort of the nodes of the resultant Jon Graph will determine a valid order for class label propagation. The class label propagation starts from the target table and proceeds outwards until either a leaf node (table) or an already extended table is reached.

To enable further automation of the classification algorithm, we use Resource Description Framework (RDF) to represent the relationship between the tables and specify the Join Graph to be used for the application. RDF is the core data model prescribed by semantic web [27]. RDF gives a simple way of representing the Join Graph through triples representing edges, and reification used to associate a label to the edge. Please note that we only represent metadata using this model, the dataset as such is not transformed to tuple form.

Further, as our intension is to use expert knowledge to improve classification accuracy, a database administrator can give additional directives on the visualized RDF file to include hidden data semantics to enhance the Join Graph. For example, "Employee-Id" in Table E is same as "EmpId" in Table F. It can also be used to specify the extent of dependency among the relational tables or data attributes. Such information enables informed pruning of the Join Graph before class label propagation. The expert can further use the inference mechanism supported over RDFS to transfer a directive specified on one table to other similar tables by just specifying them as *subclasses* of the former [28]. We believe this facility would reduce the effort needed to capture expertŠs knowledge and semantics of the database design. Another directive that we have used in our implementation is to specify only a subset of relevant attributes for the classification tasks. This is described in the context of the TPCH benchmark in section 6. While RDF does provide some nice advantages when used to represent the database metadata, our solution does not really depend upon the use of that model for correctness or accuracy. Any other means of specifying the Join Graph for the application can also be used.

4.2 Notations and Assumptions

Below are some symbolic notations we use in the rest of this section.

- The Dataset, X contains N database tables, $X_i, i = 1, ...N$
- T_i is a transformed database view of X_i and includes the (possibly propagated) class label attribute.
- The configuration of a classification application is represented with a Join Graph, a labeled directed graph. The vertex set contains, N vertices, each table corresponding to a vertex.
- H_i a classifier for table T_i appropriately chosen based on the characteristics of the table data.
- D_i is the number of dependent database tables of T_i as per the Join Graph, i.e., $D_i = \{T_j : edge < T_i, T_j > \in E\}$
- The target attribute takes K distinct class labels, $C_k, k = 1, ...K$
- P_{C_k} represents the prior probability for class label C_k over the aggregated/fused dataset. This can be got

by a single SQL query on a mega-view over original database tables. Alternatively, for cases where such a query is not feasible, either due to complex interlinkages or privacy reasons, one can make an informed subjective guess based on the application under consideration.

- $P^i_{C_k}$ represents the class priors obtained from a single transformed table, T_i.
- $P(C_k|T_i)$ is the posterior probability of class label C_k just based on the data attributes of T_i got by using classifier H_i.

4.3 Proposed Algorithm

We now describe the overall algorithm for predicting the class label of the test records using the transformed tables got by class label propagation described in section 4.1. Algorithm NBSplit-Train will be used for the training phase of the database classification and Algorithm NBSplit-Test is used for classifying the test samples.

4.3.1 Algorithm NBSplit-Train:

Input: Existing database, $X = \{X_i, i = 1, .., N\}$
Output: Classifiers H_i ; Class Priors $P^i_{C_k}$, P_{C_k}
Steps:

1. *Read database schema of base tables in X and create configuration RDF file which the application expert can edit.*
2. *Read the modified RDF file and create the Join Graph $< V, E >$ for the application*
3. *$T_r = X_r$, where $X_r \in V$ is the vertex corresponding to the target table*
4. *Starting at vertex X_r, determine a valid edge order, $E' \subseteq E$ with a breadth first search on the Join Graph.*
5. *For every edge, $(X_p, X_q) \in E'$ with label $< l_p, l_q >$: Create T_q, an extended view of X_q, by joining Tables T_p and X_q with clause $T_p.l_p = X_q.l_q$.*
6. *Compute priors from fusion table, $P_{C_k}, k = 1, \ldots K$*
7. *Training: For each table view, $T_i, i = 1, \ldots N$*
   - *Compute priors, $P^i_{C_k}, k = 1, \ldots K$*
   - *Build classifier H_i from the training set T_i*

Algorithm NBSplit-Train handles the preparation and training phase of the classification solution. The database schema is directly got from reading the meta-data of the database. As described earlier, we represent the relationship between the tables and the Join Graph to be used for the application using RDF. Any other means of specifying the Join Graph for the application can also be used. In which case, steps 1 and 2 can be appropriately modified.

In step 3 and 4, we determine the right order to propagate class labels using a breadth-first traversal over the Join Graph. We now have a list of directed edges each containing a label, which is a pair of attributes (can be set of pairs if necessary). For the running example, the tuples would be $< UNo, UNo >$ on one edge and $< ENo, Author >$ on the other edge. These pairs actually determine the Śjoin clauseŠ to create the extended view of the destination node detailed in section 4.1. In the determined order, the class labels are propagated to get a set of modified tables, $T_i, i = 1, \ldots N$, each T_i having the target attribute in addition to the original attributes of X_i. This is done in step 5.

Step 6 computes the class priors derived from the fusion table as they are used in the final equation for aggregated probability. However, as described in section section 4.2, one could safely assume equal priors or use an informed guess for simplicity and skip this step completely. Step 7 is the training step for the first level classification. Any traditional classification algorithm that can give a confidence measure along with its prediction can be used for classifying the training sets. In fact, different classification algorithms can be used for different tables based on the attribute distribution. Even simple Naive Bayes classification at the first level can be used if the attributes within a table though related, are not correlated with respect to each other. A handle to the set of classifiers built in this step, $H = \{Hi, i = 1, ..N\}$ are returned from this training phase. The class priors obtained from the transformed tables are also computed through statistical analysis.

4.3.2 Algorithm NBSplit-Test:

Input: Database X, Classifiers H_i ; Class Priors $P^i_{C_k}$, P_{C_k}, E'
Output: Predicted class label L for each test sample.
Steps:

1. *Create a test view Q_r of the target table, X_r to extract the required test record*
2. *For every edge $(X_p, X_q) \in E'$ with label $< l_p, l_q >$: Create Q_q, a filtered view of X_q, using a JOIN of Q_p and X_q with join clause $Q_p.l_p = X_q.l_q$*
3. *For each test record in table, Q_i, $i = 1..N$ Compute table-level posterior probabilities, $P(C_k|T_i)$ for each class $k = 1..K$, using the corresponding earlier trained classifier, H_i*
4. *Now traverse the Join Graph in depth first order to determine a table order for aggregation, V'.*
5. *For each node $X_j \in V'$, recursively compute scaled posterior using the following product over itself and its dependent tables, D_i*
   $P(C_k|T_j) = \frac{\{P_{C_k} \prod_i P(C_k|T_i)\}}{\prod_i P^i_{C_k}}$
6. *Classify the test sample to belong to class L when the target table is reached in the recursion. L is given by the following equation:*
   $L = argmax_k\{\frac{P_{C_k} \prod_i P(C_k|T_i)}{\prod_i P^i_{C_k}}\}$

The first step of Algorithm NBSplit-Test is to extract the required test record into a database view, which is used to extract the related attribute values of the test example in step 2. Creation of these new sets of filtered views is required for two reasons. Firstly, this is an simple way to handle one-to-many relations that may need to classify the entities based on multiple relevant records of related tables. Secondly, from a practical aspect, the traditional table-level classifiers used on single tables would require the same subset of attributes to be given in the test examples too Ű which this step provides in a simplistic fashion.

In step 3, we compute the table-level posterior probabilities by invoking the earlier trained classifiers (H_i) for each table i, described in Algorithm NBSplit-Train. It may be noted that in cases of one-to-many relationship, where multiple test examples will be passed on some individual tables levels, a decision based on maximum votes may be employed. Alternatively, we could assume independence of these relationships as well and compute the joint probability based on all test cases as a product of posteriors on individual test examples. The final decision is based on the class that gets the highest computed aggregated posterior probability at the root of the Join Graph. In particular, for two class classification, that is
Decide C_1 if

$$\frac{\{P_{C_1} \prod P(C_1|T_i)\}}{\prod P^i_{C_1}} \geq \frac{\{P_{C_2} \prod P(C_2|T_i)\}}{\prod P^i_{C_2}}$$

5. A SIMPLE PROOF

We now give a simple proof for the following aggregation equation that is recursively applied in steps 5 and 6 of the algorithm NBSplit-Test.

$$L = argmax_k\{\frac{P_{C_k} \prod_i P(C_k|T_i)}{\prod_i P^i_{C_k}}\} \quad (1)$$

The proof is based on induction over depth of the Join Graph. For simplicity, we assume binary classification. We therefore, would like to prove the decision rule
Decide C1 if

$$\frac{\{P_{C_1} \prod P(C_1|T_i)\}}{\prod P^i_{C_1}} \geq \frac{\{P_{C_2} \prod P(C_2|T_i)\}}{\prod P^i_{C_2}} \quad (2)$$

The basis condition is proved by reducing decision condition (5.2) to Bayes Rule. It can be trivially seen that , $\prod_{1 \leq i \leq r} P^i_{C_k} = P_{C_k}$ for $r = 1$ and $k = 1, 2$

and

$\prod_{1 \leq i \leq r} P(C_k|T_i) = P(C_k|T)$ for $r = 1$ and $k = 1, 2$

In effect, condition (5.2) reduces to Bayes condition:
Decide C_1 if

$$\frac{\{P_{C_1} * P(C_1|T)\}}{P_{C_1}} \geq \frac{\{P_{C_2} * P(C_2|T)\}}{P_{C_2}} \quad (3)$$

The above basis condition and decision rule holds when depth of the Join Graph is 0. This occurs when the number of nodes (tables) in the Join Graph (database) is just one or even at the leaf node of the Join Graph.

Now consider an intermediate node, R in the Join Graph with a node having r successors (we are looking at a table R which can potentially join to r other tables) and with a depth of d. Without loss of generality, figure 2 for our running example could represent one such intermediate node R (table Researcher) which is dependent on tables J (journals) and U (Unit). Our induction is on the depth of the graph. So by induction hypothesis, the decision rule holds for table U and J along with their descendants since the depth of both U and J is less than d. So, there exists a classifier H_u that can classify a test sample distributed across U and its descendants based on rule (5.2). This classification will give the same decision as that given by fusing all attributes of U and those of its descendants. Similar classifier exists for table J and its descendants.

For proving the induction rule, we can therefore consider tables U and J to be flat (fused) tables with all the attributes of itself and its descendants. We start this part of the proof by assuming that the attributes of R, U and J are all mutually independent so that we can apply Naive Bayesian rule across the further fused tables R, U and J. Let us call this hypothetical table as X. So, we get

$$\begin{aligned} P(X|C_1) = \quad & P(r_1|C_1)P(r_2|C_1)\dots P(r_r|C_1) \\ & P(u_1|C_1)P(u_2|C_1)\dots P(u_u|C_1) \\ & P(j_1|C_1)P(j_2|C_1)\dots P(j_j|C_1) \end{aligned} \quad (4)$$

We now look at individual tables and apply Naive Bayesian rule to get the posterior probabilities. Applying the NB rule to table R, for example gives

$$\begin{aligned} P(C_1|X_R) = \quad & \frac{P(X_R|C_1)P^R_{C_1}}{\sum_i P(X_R|C_i)P^R_{C_i}} \\ = \quad & \frac{P(r_1|C_1)P(r_2|C_1)\dots P(r_r|C_1)P^R_{C_1}}{\sum_i P(X_R|C_i)P^R_{C_i}} \end{aligned}$$

where X_R represents the attributes of test sample coming from R and $P^R_{C_k}$ represents the prior of class k with just the attributes of table R.

$$P(r_1|C_1)P(r_2|C_1)\dots P(r_r|C_1) = \frac{P(C_1|X_R) * \sum_i P(X_R|C_i)P^R_{C_i}}{P^R_{C_1}} \quad (5)$$

Similarly, when applied on tables U and J, we

$$P(u_1|C_1)P(u_2|C_1)\dots P(u_u|C_1) = \frac{P(C_1|X_U) * \sum_i P(X_U|C_i)P^U_{C_i}}{P^U_{C_1}} \quad (6)$$

and

$$P(j_1|C_1)P(j_2|C_1)\dots P(j_j|C_1) = \frac{P(C_1|X_J) * \sum_i P(X_J|C_i)P^J_{C_i}}{P^J_{C_1}} \quad (7)$$

Substituting equations (5.5),(5.6),(5.7) in (5.4), we get likelihood of C1 in integrated data to be

$$\begin{aligned} P(X|C_1) \quad = \quad & \frac{P(C_1|X_R)\sum_i \left(P(X_R|C_i)P^R_{C_i}\right)}{P^R_{C_1}} \\ & \frac{P(C_1|X_U)\sum_i \left(P(X_U|C_i)P^U_{C_i}\right)}{P^U_{C_1}} \\ & \frac{P(C_1|X_J)\sum_i \left(P(X_J|C_i)P^J_{C_i}\right)}{P^J_{C_1}} \\ P(X|C_1) \quad = \quad & \frac{P(C_1|X_R)P(C_1|X_U)P(C_1|X_J)}{P^R_{C_1}P^U_{C_1}P^J_{C_1}} * Q \end{aligned}$$

where

$$Q = \sum_i \left(P(X_R|C_i)P^R_{C_i}\right) \sum_i \left(P(X_U|C_i)P^U_{C_i}\right) \sum_i \left(P(X_J|C_i)P^J_{C_i}\right)$$

Now the decision rule for binary classification based on integrated data will be:
Decide C_1 if $P(C_1|X) \geq P(C_2|X)$
that is,

$$\frac{P(X|C_1)P_{C_1}}{P(X)} \geq \frac{P(X|C_2)P_{C_2}}{P(X)}$$
$$\Rightarrow P(X|C_1)P_{C_1} \geq P(X|C_2)P_{C_2}$$

From the earlier equations, we get
Decide C1 if

$$\frac{P(C_1|X_R)P(C_1|X_U)P(C_1|X_J)P_{C_1}}{P_{C_1}^R P_{C_1}^U P_{C_1}^J} \geq \frac{P(C_2|X_R)P(C_2|X_U)P(C_2|X_J)P_{C_2}}{P_{C_2}^R P_{C_2}^U P_{C_2}^J}$$

Rewriting the above equation with general terminology, we get
Decide C1 if

$$\frac{\{P_{C_1}\prod P(C_1|T_i)\}}{\prod P_{C_1}^i} \geq \frac{\{P_{C_2}\prod P(C_2|T_i)\}}{\prod P_{C_2}^i}$$

which is same as equation (5.2). Hence, the proof.

Though we initially assumed mutual independence of all attributes and considered Naive Bayes on individual tables, since the final decision is just based on the posterior probabilities from the individual tables, any other classifier that provides a confidence measure of prediction may be used at the local level. This means that we just need to assume mutual independence of attributes across two tables, the attributes within each base table may be related. This means one is not restricted to use Naive Bayes classifier on individual tables and can use any other classifier that gives a confidence measure (probability) with its decision (such logistic regression) at local level as shown in next section.

6. IMPLEMENTATION AND RESULTS

We now state some preliminary empirical results obtained after implementing the proposed algorithm for classifying multi-relational dataset benchmarks. Our implementation is in Java language and uses the Weka 3.5.7 Machine Learning package for classification algorithms on individual database tables. As mentioned earlier, the Join Graph for the databases is represented using the RDF data model [27]. We use the Jena Java library to manipulate and query the configuration file describing the Join Graph. We store the databases on PostgreSQL 8.2 on Windows XP machine.

We evaluated the classification algorithms on three types of datasets. Firstly, on a popular DBMS benchmark used to measure scalability and performance of database applications (TPCH). Secondly, a well-known single-table dataset from UCI (Lung Cancer) to prove the correctness of the algorithm and demonstrate the heterogeneity capabilities of the algorithm. Thirdly, we use a multi-relational dataset (the financial database from PKDD Cup 99) to compare with prior relational classification techniques.

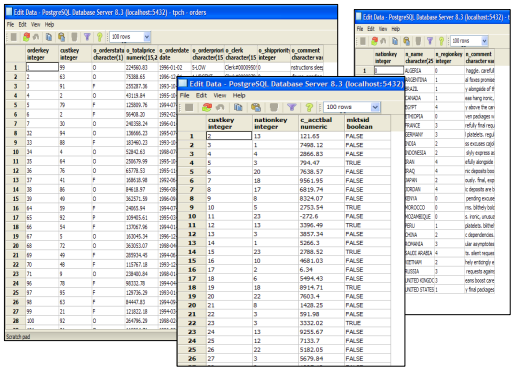

Figure 3: TPCH database

Num of records in Nation table :	99
Num of records in Customer table:	99
Num of records in Orders table:	98
Total No. records with relational data:	296
Total No. records in the fused tables:	165000
Number of training samples for NBSplit	296
Num of training samples in fused data	165000
Time for training with fused tables	**772 msec**
Time for training with relational tables	**20 msec**

Table 1: Training Time for TPCH dataset

6.1 TPCH benchmark

TPCH is a synthetic database simulating a real life data source of Customer Relationship Management. We used the CUSTOMER, ORDERS and NATION database tables shown in Figure 3. The problem was to predict whether a given customer was likely to buy an household item or not. Our aim for using this benchmark was to show the practical aspect of our solution. Firstly, use of RDF to specify the subset of data attributes of interest was very useful here. For example, the ORDERS table originally had 9 attributes of which we found that only four attributes were meaningful for the classification. Specifying this subset in RDF helped in auto generation of the views. In fact, our solution is so automated that just a change in the input RDF file (see Figure 4 for a sample) is sufficient to execute the classification task on a completely new set of database tables (once on researcher database and next on TPCH, for example).

Secondly, we wanted to experiment with the amount of overhead one can incur by performing the mega join of all tables and applying simple Naive Bayes algorithm over the fused dataset. For this, we chose a smaller TPCH dataset with just 100 records in each table and computed the training time of the algorithm for both fused Naive Bayes and our Split-NB algorithm. We used only Naive Bayes classifier on individual tables here. Table 1 demonstrates the dramatic improvement in the training time with the TPCH dataset. As seen in the number of training samples and the training time of the dataset, our approach scales over fused dataset in orders of magnitude.

6.2 Lung-Cancer Dataset

We tested the correctness and accuracy of the classifica-

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
    <!ENTITY owl  "http://www.w3.org/2002/07/owl#" >
    <!ENTITY xsd  "http://www.w3.org/2001/XMLSchema#" >
    <!ENTITY rdf  "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
    <!ENTITY dm  "http://example.org/datamining#">
  ]>

<rdf:RDF
  xmlns ="&dm;"
  xmlns:dm      ="&dm;"
  xml:base  = "&dm;"
  xmlns:owl = "http://www.w3.org/2002/07/owl#"
  xmlns:rdf = "http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:rdfs= "http://www.w3.org/2000/01/rdf-schema#"
  xmlns:xsd = "http://www.w3.org/2001/XMLSchema#">

<owl:Ontology rdf:about="datamining">
</owl:Ontology>

<dm:Classification rdf:ID="work">
  <dm:targetAttr rdf:resource="#mktsid" />
  <dm:targetTable rdf:resource="#customer" />
</dm:Classification>

<dm:table rdf:ID="customer">
  <dm:attr rdf:resource="#c_acctbal" />
  <dm:attr rdf:resource="#mktsid" />
  <dm:pkey rdf:resource="#custkey" />
   <dm:attr rdf:resource="#custkey" />
  <dm:fkey rdf:resource="#nationkey" />
  <dm:ftable rdf:resource="#nation" />
</dm:table>

<dm:table rdf:ID="orders">
  <dm:attr rdf:resource="#o_totalprice" />
  <dm:attr rdf:resource="#o_orderpriority" />
  <dm:pkey rdf:resource="#orderkey" />
  <dm:attr rdf:resource="#orderkey" />
  <dm:fkey rdf:resource="#custkey" />
  <dm:ftable rdf:resource="#customer" />
</dm:table>

<dm:table rdf:ID="nation">
  <dm:pkey rdf:resource="#nationkey" />
  <dm:attr rdf:resource="#nationkey" />
  <dm:attr rdf:resource="#n_regionkey"/>
</dm:table>
</rdf:RDF>
```

Figure 4: RDF File used for TPCH

tion results using a standard single-table dataset from UCI. For this, we artificially loaded the 58 attributes of the Lung Cancer dataset into 7 tables and executed the proposed algorithm over this distributed data. For simplicity, we assumed a Star schema. We compared the results of classification in this scenario with those of the original fused dataset. The predictions got by both these procedures were exactly the same when we used Naive Bayes classifier at local level (since the aggregation is also based on Naive Bayes condition). This demonstrates the correctness of our algorithm. We further tested our algorithm with different but common single-table classifiers, both over the fused data as well as first-level classifiers in our algorithm. The results are shown in the Radar graph shown in Figure 5. Though the accuracy results heavily depend upon the characteristics of the dataset and the way we partition it into multiple tables, we believe this exercise proves the effectiveness of our collation algorithm for diverse table-level classifiers.

6.3 Financial Database

Lastly, we tested our solution on the financial database from PKDD CUP 99, shown in Figure 6. Here, we tried multiple common classifiers on individual tables and chose the best classifier for every table. This chosen best classifier per table was then used to create a heterogeneous classifier which performed very well (only first level dependencies were considered). We chose a voting scheme to handle the 1-many relationships resulting in a single prediction for every loan-id.

We then compared our results with that of Cross Mine [2]

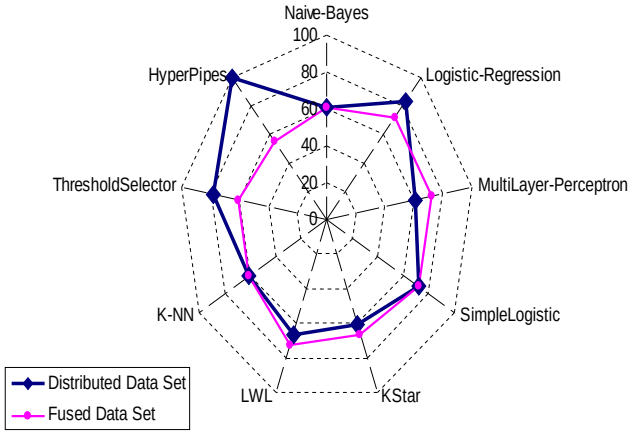

Figure 5: Accuracy for different 1-table classifiers

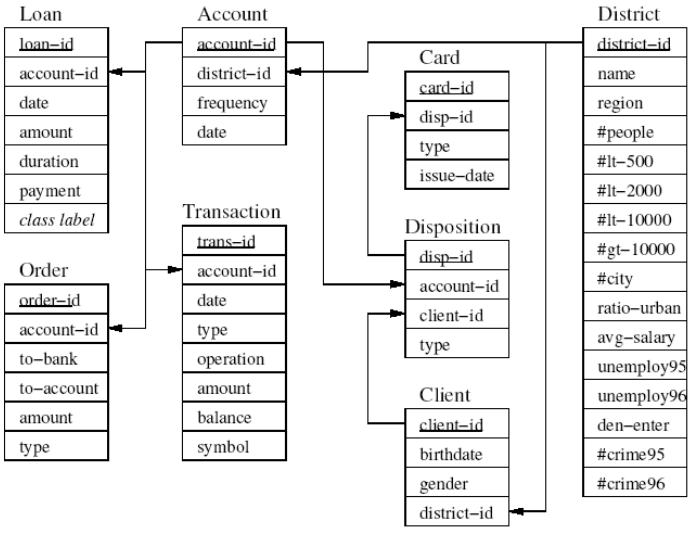

Figure 6: Financial Database

multi-relational classifier based on meta learners, we found that our technique performs much better. We also found that elimination of non-contributing tables results in much better results as well - proving the benefit of using hidden semantics of schema design (relevant feature sets are selected automatically). Consistently across multiple classifiers, the Transaction table was causing negative effects on the classification accuracy. Similarly, inclusion of Account and Order tables increased the accuracy results, as opposed to using just Loan table, showing the need for including other related tables for accurate prediction. Results are shown in Table 2.

7. CONCLUSIONS

We proposed a scalable, two-phase classification algorithm for classifying relational data. Our main premise is that since most databases are well designed by experts, a lot of semantics is already implicit in the database schema. Us-

Accuracy from Cross Mine Algorithm	85
Best accuracy from just Target Table (Loan)	77.5
NB-Split with Multi-layerPerceptron on all tables	92.5
NB-Split with Multi-layer Perceptron without trans	95
NB-Split with LibSVM on all	100
NB-Split with Heterogeneous Classifiers with all	90
NB-Split with Hetero Classifiers without trans	95

Table 2: Classification Accuracy for Financial DB

ing this semantic information, we proposed a new recursive aggregation technique to collate heterogeneous classifiers applied at individual tables. We have demonstrated obvious improvement in classification training time for standard DBMS benchmark, TPCH. We proved that there is no loss of accuracy due to our recursive aggregation and also showed the benefits of our approach over prior efforts. As shown, this procedure works even when different classifiers are used on different tables at the first level.

We would like to extend this technique to include selection of the right classifier at the table level, which is manual for now. The generated Join Graph can be further pruned to improve the classification time, by eliminating tables that may not really contribute much to the overall classification task. For this, we plan to associate an entropy metric with individual database tables and select the right subgraph from the Join Graph that minimizes the information loss. We would also like to explore other ways of utilizing the additional hidden semantics in the data.

Our intension is to make classification of databases a reality. We have therefore used existing database features such as SQL views, meta-data from the database schema, generation of test data and use of expert-editable RDF file. Use of RDF file for storing the configuration of the application really helps in automating the classification task. Going forward, we would like to test this algorithm on real enterprise databases with complex relationships. In particular the domain of predicting the health of a server based on multiple IT parameters [30] is one useful application we are addressing .

8. ACKNOWLEDGEMENTS

We acknowledge the help of Ms Pooja Yadav, IISc in the initial implementation of this solution.

9. REFERENCES

[1] Homgyan Liu, Xiaoxin Yin, Jiawei Han, "An Efficient Multi-relational Naive Bayesian Classifier Based on Semantic Relationship Graph", ACM MRDM, 2005

[2] Xiaoxin Yin, Jiawei Han, Jiong Yang, Philip S Yu, "CrossMine: Efficient Classification Across Multiple Database Relations", ICDE 04, 2004

[3] Hongyu Guo, Hema L Victor, "Mining Relational Databases with Multi-View Learning", ACM MRDM 05.

[4] Michelangelo Ceci, Annalisa Appice, Donato Malerba, "Mr-SBC: a Multi-Relational Naive Bayes Classifier", LNCS, PKDD 2003

[5] Peter A. Flach, Nicolas Lachiche, "Naive Bayesian Classification of Structured Data", Machine Learning, 2004

[6] Xindong Wu, Chengqi Zhang, "Database classification for multi-database mining", Information Systems archive, March 2005, ISDN:0306-4379

[7] Huan Liu, Hongjun Lu, Jun Yao, "Toward Multidatabase Mining: Identifying Relevant Databases", IEEE Transactions on Knowledge and Data Engineering, July 2001

[8] Shichao Zhang, Xindong Wu, Chengqi Zhang, "Multidatabase mining", IEEE Comput. Intell. Bulletin 2003

[9] Jing-Feng Guo, Jing Li, "An Efficient Relational Decision Tree Classification Algorithm", ICNC 07

[10] Rayner Alfred and Dimitar Kazakov, "Aggregating Multiple Instances in Relational Database Using Semi-Supervised Genetic Algorithm-based Clustering Technique", ADBIS 2007

[11] Ben Taskar, Eran Segal, Daphne Koller, "Probabilistic Classification and Clustering in Relational Data", Stanford University, IJCAI 2001

[12] Blum A, Mitchell T M, "Combining labeled and unlabeled data with co-training", COLT 98

[13] Ghani R, "Combining labeled and unlabeled data for multiclass text classification", ICML 2002

[14] Paolo Giudici, "Applied Data Mining, Statistical Methods for Business and Industry", 2003

[15] Breiman L, "Bagging Predictors", Machine Learning, 24 (2), 1996, 123-140.

[16] Freund Y, Schapire R, "Experiments with a New Boosting Algorithm", ICML 1996.

[17] David H Wolpert, "Stacked Generalization", Neural Network, 5, 1992, 241-259.

[18] S Muggleton, "Inductive Logic Programming", Academic Press, New York, NY, 1992.

[19] Blockeel H, DeRaedt L, Ramon J, "Top-down induction of logical decision trees", ICML 98

[20] J R Quinlan, R M Cameron-Jones, "FOIL: A midterm report", Madison, WI, 1998 European Conf. Machine Learning, Vienna, Austria, 1993.

[21] Getoor L, Friedman N, Koller D, Taskar B, "Learning probabilistic models of relational structure", ICML 2001.

[22] Domingos P, Pazzani M, "On the optimality of the simple bayesian classifier under zero-one loss", Machine Learning, 29(2-3), pp. 103-130, 1997.

[23] R Elmasri, S B Navathe, "Fundamentals of Database Systems", Addison-Wesley, 4th ed, 2004

[24] Jennifer Neville, David Jensen, "Iterative Classification in Relational Data", AAAI 2002

[25] S Slattery, T Mitchell, "Discovering test set regularities in relational domains", ICML 2000.

[26] D Koller, A Pfeffer, "Probabilistic frame-based systems", AAAI 1998.

[27] Semantic Web, W3C, http://www.w3.org/2001/sw/

[28] RDFS, Resource Description Framework, Schema Specification 1.0, W3C , March 2000

[29] B Liu, W Hsu, Y Ma, "Integrating classification and association rule mining", In Proc of KDD, 1998

[30] G. Manjunath, "Semantic Web for Enterprise Data Integration and Service Composition", SENOPT, HiPC, 2007