HeteroClass: A Framework for Effective Classification from Heterogeneous Databases

CS512 Project Report

Mayssam Sayyadian

May 2006

Abstract

Classification is an important data mining task and it has been studied from different perspectives. Recently multi-relational classification algorithms has been studied due to many real-world applications. However, current work has generally assumed that all the needed data to build an accurate prediction model resides in a *single* database. Many practical settings, however, require that we combine tuples from *multiple* databases to obtain enough information to build appropriate models for the classification task. Such databases are often *autonomous*, and *heterogeneous* in their schemas and data. In this project, we describe HeteroClass, a framework for effective classification from heterogeneous databases. First we show how to combine schema matching and structure discovery techniques to find approximate foreign-key joins across heterogeneous databases. Such joins are critical for producing accurate models that capture the required information that spans across multiple databases and relations.

We then develop an ensemble construction algorithm that exploit the joins – discovered automatically across the databases – to enable effective classification across the distributed data, using specialized classifiers on more homogeneous regions. Since in multi-database settings, we have to deal with attribute instability, i.e., data with various characteristics and distributions, our algorithm guarantees that the created ensemble is diverse enough to cope with these challenges. Our extensive experiments over two real-world data sets show that (1) our classification framework outperforms single database and its counterpart multi-database algorithms in terms of accuracy, and (2) our approach manages to produce links across the heterogeneous databases, with no need for human reconciliation of the different databases.

1 Introduction

Classification is a well-studied and effective way of data analysis that can be used to extract models describing important data classes or to predict future data trends. So far, the literature on classification of data over relational databases has assumed that all relevant information for building the model and predicting labels is present within a *single* database. However, the most acute information management challenges today stem from organizations

relying on a large number of diverse, but interrelated data sources. These scenarios happen in enterprise and government data management, digital libraries, "smart" homes and personal information management system. In such practical settings assumption does not hold and we must combine tuples from multiple databases to obtain enough information to build an accurate prediction model. Furthermore, the databases of interest are often autonomous and heterogeneous in the schemas that they support and in the data that they contain. The following example illustrates these issues:

Example 1.1 *Consider an academic organization, such as University of Illinois. The dean of the school wants to understand which graduates of the school are more probable to donate money to the school in future. This is an interesting information need, in which data mining techniques such as classification is supposed to be helpful. Intuitively, the dean suspects that the degree of the graduates (B.Sc., M.S., PhD, etc.) is one useful predicate in building such a classifier, so they need the information store in the database of the "Office of Admission and Record". He also suspects that the departmental information about the students (such as a student's adviser), student's courses, and the information about communities that the students used to be a member of, are also useful and beneficial in building a more accurate and useful prediction model. So they have to combine the various databases in various departments (in multiple databases) to build the classifier on top of all of them.*

Other examples of the need for classifying data that is scattered across multiple databases arise naturally. For instance, suppose that two companies join forces to develop a product and want to predict some information need about their competitors. Again, they need to combine their databases to build a useful model. Yet as another example, consider an enterprise that is willing to know which customers will be using both traveling and dining services of that company. As another important example, assume that we want to build a model to predict which movie scenes are captured outdoor. This requires combining multimedia and spatial databases as well as annotation databases (e.g. movie scripts, etc.).

As the above examples show, the ability to perform classification over multiple databases is important in many practical settings, and will become increasingly so as the number of such databases grows. In this project, we describe HeteroClass, a solution to the classification problem over heterogeneous databases. As we will see, developing HeteroClass raises significant challenges:

Automatic Integration: The first challenge to unified model over heterogeneous databases is how to integrate database schemas and data. Databases for the potentially dynamic scenarios that we consider have often not been integrated and exhibit *semantic heterogeneity* [24]. For example, employee names can be referred to as emp-name and name, id can refer to different types of id, and Michael Smith can be referred to as "Michael Smith" and "Mike D. Smith" in different databases. *Manually* integrating such heterogeneous databases is well known to be difficult, and might take weeks or months to accomplish [24, 9]. Manual integration is thus not appropriate for our context, where answers often must be obtained quickly.

In addition, many classification tasks express *ad-hoc, short-term* information needs, and hence they require only a *temporary* assembling of several databases. Note that this heterogeneity problem does not arise in the single-database scenarios considered so far in the

literature. To address this heterogeneity problem, we develop a solution that *automatically* discovers approximate foreign-key joins across heterogeneous databases, since such joins are critical for playing the role of a bridges that connect multiple relations. We employ a combination of structure discovery and schema matching methods that empirically outperforms current join discovery algorithms.

Attribute Instability: As explained, many of such large-scale data analysis problems involve an investigation of relationships between attributes in heterogeneous databases, where different prediction models can be responsible for different regions. However, large data sets very often exhibit attribute instability, such that the set of relevant attributes is not the same through the entire data space. This is especially true in spatial databases, where different spatial regions may have completely different characteristics. This is also true in organizations and enterprises, where different departments capture different pieces of information about a concept.

To address these issues we develop HeteroClass framework that employs best-effort data integration techniques to handle heterogeneity of the databases. We also use ensemble of classifiers where each member of the ensemble is an expert classifier for some part of the data. We propose an algorithm that ensures this diversity in the ensemble will not degrade the whole accuracy of the committee.

In summary we make the following contributions:

- We define the problem of classification across heterogeneous databases. To the best of our knowledge, this is the first solution to this important problem, considering the intrinsic challenges arising in this context such as heterogeneity and attribute instability.
- We develop an ensemble based classification framework that guarantees to improve the accuracy of its base learners. We also use best-effort integration technique that discovers foreign-key joins across databases in a robust fashion, and show how data integration techniques can help data mining.
- We report extensive experimental results over real-world data sets. Our results suggest that our ensemble construction method produces high-quality prediction models and improves the accuracy of base learner, with no need for human reconciliation of the different databases.

The rest of this report is organized as follows: In Section 2 we review the main challenges that arise in the context of multi-database classification. Then, in Section 3 we introduce our key observations and HeteroClass approach to this problem. We review related work in Section 4 and next, in Section 5, we describe HeteroClass framework. We report empirical evaluation in Section 6 and conclude in Section 7.

2 Challenges

The traditional solution to this problem is constructing data warehouses, which involves data cleaning, data integration, and data transformation as a preprocessing step for data mining.

Not only this is an intensive time and resource consuming task, but also in many scenarios it is inefficient and even impossible according to the following observations:

- Studying heterogeneous databases has been an active research for long time. A key challenge of heterogeneous databases is how to deal with semantic heterogeneity presented by the multiple autonomous databases. Techniques for schema analysis, translation, and integration attempt to solve the semantic heterogeneity at the schema level. Consequently integrating data from these heterogeneous sources is difficult, as much research on schema matching and schema mapping has shown. However in many data mining tasks such as building prediction models, we do not need to fully integrate the data sources. Best effort data integration techniques, although approximate and not accurate, can help in extending data mining algorithms to multiple-database settings.

- Schema level analysis may touch just one level of semantic heterogeneity problem. Many heterogeneous databases, even being analyzed and transformed into a federated database, may still have problems in communicating effectively with each other. A deeper level analysis of semantic heterogeneity, called data-level heterogeneity, which is the analysis of database contents should be taken into consideration. Data cleaning, data transformation and multi-level data analysis has been the focus of a lot of research efforts in past decade that deal with semantic heterogeneity at data level.

- Many large-scale data analysis problems involve an investigation of relationships between attributes in heterogeneous databases, where different prediction models can be responsible for different regions. In addition, large data sets very often exhibit *attribute instability*, such that the set of relevant attributes is not the same through the entire data space. This is especially true in spatial or multimedia databases, where different spatial regions or multimedia sections may have completely different characteristics. This is also true when the underlying data for an organization is distributed either horizontally or vertically through multiple (overlapping) data sources.

- In many scenarios data sources join and leave the network frequently. Most obvious example of this is peer-to-peer data sharing systems. As another example consider companies such as Yahoo Shopping, eBay, Amazon, etc. that have a service to list the inventory of external and personalized stores. For such companies many external stores (with their underlying databases) may join and leave their network of databases. In many cases, autonomous databases evolve and change both at the data and the schema level over time.

- Privacy and security is a challenge when we deal with multiple autonomous disparate data sources. In many settings only a limited querying permissions are granted to access a database. In many other cases, only some (vertical/horizontal) partitions of the database are available for external data consumers. In general, in many real-world scenarios it is not possible to extract data from an autonomous database (that for example belongs to the customers of a company) and move that into a data warehouse.

- Performance is always an issue when we deal with large amounts of data. In general moving large bulks of data across network is not desirable, even if possible. So algo-

rithms try to evaluate and explore data locally and avoid moving data in the network as long as it is possible. Communication cost is the performance bottleneck of most distributed algorithms.

This motivates investigating new class of data mining methods that deal with data residing in various, physically separate and heterogeneous data sources. As a first step we consider classification as one of the main data mining tasks.

3 Observations and Our Approach

Data Level Heterogeneity Our first observation is that data values in multiple heterogeneous databases do not agree all the time. For example one database will refer to "John Smith" as "J. Smith", while the other may keep track of it as "J. K. Smith". More importantly, in relational databases each table has a key attribute, e.g. tuple-id and these attribute and their values play an important role in building join paths across different relations. But these keys are useless in our context, because when we come across a new table that belongs to a different database, the two tables often do not share the same id space. As a result, we cannot find any join path across two separate databases that involves tuple-id. We argue that we need to employ an approximate value matching technique for our query processing that compares data values. We will use the state of the art approximate value matching techniques using q-grams.

Schema Level Heterogeneity We observe that even if possible, it is very labor intensive and error prone to consolidate all schemas into one universal schema, specially if we want to consider unseen or evolving schemas. Moreover, we observe that we do not need every attribute in every schema and building a large and universal mediated schema is not even required. Previous work in multi-relational classification show that join paths and links among relations are what we need in the training and testing phase. So we suggest an automatic approach in which we automatically explore links that are required for building the prediction model. We focus on discovering soft-functional dependencies and key-foreign keys among the databases, because they carry more semantic weight and they are more useful in building a classification model.

We employ a novel combination of schema matching and structure discovery methods that outperform current join discovery algorithms. The idea of most structure discovery algorithms that explore join paths is based on computing statistical synopses for attributes and then using them in finding "joinable" attributes. However in our setting many columns will be discovered joinable because we need to use an approximate value matching technique based on q-grams. We use schema matching as a tool that finds corresponding attributes among different databases, in conjunction with regular structure discovery methods to solve this problem.

Heterogeneity at the Structure/Format of Data Sources In many practical settings, the format and structure of the underlying data sources that capture the same concept are different considerably. For example one data source may provide data in XML and

hierarchical format, another data source may capture related information in a relational database. Moving data from one format to the other, without loosing structural information, is considered difficult and costly. In order to find appropriate links to connect these sources, we need generic methods for consolidating the schema-level and structural heterogeneity. We employ a generic graph-based model matching algorithm, *Similarity Flooding*, that given two graph representations of any underlying model, provides useful information about matching links across those data sources.

Diversity and Attribute Instability As we discussed in many large-scale data analysis problems involve an investigation of relationships between attributes in heterogeneous databases, where different prediction models can be responsible for different regions. In addition, large data sets very often exhibit attribute instability, such that the set of relevant attributes is not the same through the entire data space. On the other hand we know that ensemble methods that combine the decisions of multiple hypotheses are some of the strongest existing machine learning methods, in the settings in which the members of the ensemble are diverse.

We employ the ensemble of classifiers approach to build multiple accurate and specialized classifiers in more homogeneous regions so as to improve the global accuracy of the whole ensemble over the whole data sources. We propose a method for building the ensemble that guarantees improvement in accuracy (in the training phase), while diversifies the member classifiers. It also uses the links across databases whenever available, to build more general models.

4 Related Work

Classification and prediction has been the focus of a lot of research in past decades[11] from many different perspectives. A recent trend of research in past few years has focused on multi-relational data mining. In particular classification and clustering across multiple relations is studied in [30, 31].

However there is little work on data mining in a heterogeneous environment. To the best of our knowledge [29] is the only piece of work that addresses the problem of classification of data in multiple heterogeneous databases. [29] proposes a regression-based method for predicting the usefulness of inter-database links that serve as bridges for information transfer and also suggests an *economical* strategy in their rule-based classification algorithms, however this works assumes the heterogeneity problem of the databases are solved and they only tune classification algorithms for better performance. Our HeteroClass framework is different from this work in that we address the intrinsic problems that arise from heterogeneity of databases. In particular we address data level and schema level heterogeneity. Our approach is also different in that we first build specialized and more accurate prediction models in local databases and then merge them to build a global prediction model using an ensemble approach. In [23] methods for linking and mining massive heterogeneous databases are suggested. They loosely define fields that allow free style verbatim entries. Authors developed an interestingness measure based on non-parametric randomization tests, and use that for mining potentially useful relationships among variables by solving the record linkage prob-

lem. [33] Addresses the problem of peculiarity mining in a multi-database setting, however heterogeneity at the schema level is not addressed in this work and our work is different in that we exploit rich amount of meta-data available in the databases.

Other distributed data mining methods are also discussed in literature, but they never consider heterogeneity of data sources. In general there are two types of distributed data: (1) horizontally partitioned data, and (2) vertically partitioned data. Either way of distribution divides the rows or columns of a table into different parts. Distributed data mining approaches for horizontally partitioned data include meta-learning [3] that merges models built from different sites, and privacy preserving techniques including decision tree [19] and association rule mining [17]. Those for vertically partitioned data include association rule mining [27] and k-means clustering [28]. None of these methods consider heterogeneous databases and challenges in such settings.

There is also a large body of work on schema matching (e.g., [6, 8, 16], see [24] for a survey) which develop semi-automatic matching tools. Typically, such a tool is applied to find matches, then a user manually verifies and corrects the predicted matches before querying can be carried out [24, 6]. Our work shows practical settings where automatic schema matching can have immediate benefits, and where it is not realistic to assume that the user will have time or expertise to manually verify the matches.

A wealth of techniques have also been developed to match data instances (e.g., [4, 20, 1, 25, 10, 14]). We currently employ the solution proposed in [10] because it is directly implementable using SQL queries in databases, and thus can be naturally and quickly folded into our solution.

5 HeteroClass Framework

In this section we describe our HeteroClass framework. We begin by explaining the architecture and the main components in this section. Then in Section 5.1 we describe the join discovery module that combines schema matching with state-of-the-art join discovery techniques. Then in Section 5.2 we describe the ensemble builder module, that builds diverse, and accurate ensemble of classifiers on top of the data sources, using the links provided by join discovery module.

5.1 Discovering Joins Across Databases

This module discovers a set of foreign-key joins across the given databases. In developing this module, we found that current join discovery algorithms (e.g., [15, 5, 2]) can be significantly improved by combining them with schema matching techniques (see Step 4 below).

Consider two tables U and V that belong to different databases. Our goal is to find all foreign-key joins in V that reference a key of table U. For this, we first find all approximate keys in U, since they will participate in any foreign-key joins that we discover. Then, we consider each key of U individually, and identify any attribute sets in V that could be meaningfully joined with the key. After this, we generate candidate foreign-key joins. Finally we only keep candidates that are "semantically correct", as we discuss below:

1. Finding keys in table U: Sometimes table U might include keys that are not helpful for participating in foreign-key joins across databases. For example, an id attribute of U might be meaningless to join with a table V in some other database, because even if U and V are indeed related at the data level, V often does not share the same id space with U. Rather than discovering or exploring true keys such as id above, we focus on finding "approximate" keys in table U that help in defining approximate foreign-key joins. For this, we employ an approximate key discovery algorithm recently developed in [15] to find such keys for U.

2. Finding joinable attributes in V: Once we have found the approximate keys of U, we find attributes in V that can be joined with the attributes of these keys. Specifically, for each attribute a in an approximate key K of U, we find all attributes b in V such that a and b are *joinable*, in that they share many similar values.

Inspired by the Bellman system[5], we compute statistical synopses for attributes and then employ these synopses to quickly find "joinable" attributes for large databases. The algorithm finds approximate join paths based on set resemblance. We define the sets as the unique values in the fields. This way the resemblance of two fields is a good measure of their similarity. We find the similarity of numerical fields by exact match and for textual fields, we use substring similarity (q-grams) to alleviate data-level heterogeneity. The resemblance of two sets A, B is defined as follows:

$$\rho(A, B) = |A \cap B|/|A \cup B| \tag{1}$$

For our purpose we are more interested in computing the size of the intersection of A and B, which can be computed from the resemblance by:

$$|A \cap B| = \frac{\rho}{\rho + 1}(|A| + |B|) \tag{2}$$

We profile the number of distinct values of each field, so $|A|$ and $|B|$ is always available.

The real significance of the resemblance is that it can be easily estimated. Let $\mathcal{U}$ be the universe of values from which elements of the sets are drawn, and let $h : \mathcal{U} \rightarrow \mathcal{N}$ map elements of $\mathcal{U}$ uniformly and "randomly" to the set of natural numbers $\mathcal{U}$. Let $s(A) = \min_{a \in A}(h(a))$. Then

$$\Pr[s(A) = s(B)] = \rho \tag{3}$$

That is, the indicator variable $I[s(A) = s(B)]$ is a Bernoulli random variable with parameter ρ. As an explanation, consider the inverse mapping $h^{-l} : \mathcal{N} \rightarrow \mathcal{U}$. The function h^{-1} defines a sampling strategy for picking an element of the set A, namely select $h^{-l}(0)$ if it is in A, else $h^{-l}(1)$, else $h^{-1}(2)$, and so on. The process is similar to throwing darts at a dart board, stopping when an element of the set is hit. Let us consider $h^{-l}(k)$, the first element in the sequence which is in $A \cup B$. The chance that $h^{-l}(k) \in A \cap B$ is ρ, and is indicated by $s(A) = s(B)$.

3. Generating foreign-key join candidates: Once we have identified (approximate) keys in U and matched them against attributes in V, we identify candidate foreign-keys by

exhaustively listing all possible alignments of the key attributes in U with their joinable counterparts in V. As an example, consider a key $\{a_1, a_2\}$ in U and suppose that attribute a_1 is joinable with attribute b_1 of V, while attribute a_2 is joinable with both attributes c_1 and c_2 of V. Then we list two candidate foreign-key joins, J_1, where attributes (b_1, c_1) of V reference attributes (a_1, a_2) of U, and J_2, where attributes (b_1, c_2) of V reference attributes (a_1, a_2) of U.

4. Removing semantically incorrect candidates: Not all candidate foreign-key joins are meaningful, since current join discovery algorithms [5, 15, 2] examine only the similarity of *data values* to produce join candidates such as J_1 and J_2 in our example above. In fact, attributes may share similar values and yet not be semantically joinable, as is the case for last-name and city-name (both with string values), or electricity and water-front (both with "yes"/"no" values).

To filter out spurious candidate foreign-keys, we introduce a schema matching step that examines the database *schemas* to find semantically *related* attributes. We then keep only join candidates with semantically related attributes. For example, consider join candidate J_1, where attributes (b_1, c_1) of V reference attributes (a_1, a_2) of U. We discard J_1 if either b_1 is found not to match a_1 or c_1 is found not to match a_2 by a schema matching algorithm. Virtually any generic schema matching algorithm [24, 9] can be used in this step. We use a version of Simflood algorithm [21]. SimFlood computes a semantic distance score for any two attributes based on the similarity of their names and those of neighboring attributes. This is achieved as follows.

As a first step, we translate the schemas from their native format into graphs G_1 and G_2. As a second step, we obtain an initial mapping between G_1 and G_2 using a string matcher operator. The similarity values range between 0 and 1 and indicate how well the corresponding nodes in G_1 match their counterparts in G_2. Notice that the initial mapping is still quite imprecise. As a third step, an iterative graph matching operator is applied to produce a refined mapping between the two graphs. In [21], an iterative "similarity flooding" (SF) algorithm based on a fixpoint computation is proposed. The SF algorithm has no knowledge of node and edge semantics. As a starting point for the fixpoint computation the algorithm uses an initial mapping like what we mentioned earlier. The algorithm is based on the assumption that whenever any two elements in models G_1 and G_2 are found to be similar, the similarity of their adjacent elements increases. Thus, over a number of iterations, the initial similarity of any two nodes propagates through the graphs. The algorithm terminates after a fixpoint has been reached, i.e. the similarities of all model elements stabilize. As a last operation in the script, we selects a subset of node pairs that corresponds to the "most plausible" matching entries, normalize their score and output them as the our confidence score in the pair's semantic similarity.

We filter out any candidate foreign-key joins whose attributes do not pass the above schema matching test, and return all the surviving foreign-key joins among all relation pairs in the databases. Note that we focus on discovering "full" foreign-key joins and ignore partial matches where only some but not all of the key attributes of a relation are joinable with attributes of another relation.

5.2 Building Diverse Ensemble of Classifiers

Combining multiple models is an effective technique for improving classification accuracy. An ensemble (committee) of classifiers is a set of classifiers whose individual decisions are combined in some way (typically by weighted or unweighted voting) to classify new examples. One of the most active areas of research in supervised learning has been to study methods for constructing good ensembles of classifiers. In general, an ensemble method is used to improve on the accuracy of a given learning algorithm. We will refer to this learning algorithm as the base learner. The base learner trained on the given set of training examples is referred to as the base classifier. It has been found that in most cases combining the predictions of an ensemble of classifiers produces more accurate predictions than the base classifier [7].

In an ensemble, the combination of the output of several classifiers is only useful if they disagree on some inputs [12, 26]. We refer to the measure of disagreement as the *diversity/ambiguity* of the ensemble. For regression problems, mean squared error is generally used to measure accuracy, and variance is used to measure diversity. In this setting, it is shown that the generalization error, E, of the ensemble can be expressed as $E = \overline{E} - \overline{D}$ where $\overline{E}$ and $\overline{D}$ are the mean error and diversity of the ensemble respectively. This result implies that increasing ensemble diversity while maintaining the average error of ensemble members, should lead to a decrease in ensemble error. Unlike regression, for the classification task the above simple linear relationship does not hold. But there is still strong reason to believe that increasing diversity should decrease ensemble error [32].

5.2.1 Diversity Measure

There have been several measures of diversity for classifier ensembles proposed in the literature. In a recent study, [18] compared ten different measures of diversity. They found that most of these measures are highly correlated. In our work we use the diversity measure used in [22]. For our work, we use the disagreement of an ensemble member with the ensemble's prediction as a measure of diversity. More precisely, $C_i(x)$ is the prediction of the i-th classifier for the label of x; $C^*(x)$ is the prediction of the entire ensemble, then the diversity of the i-th classifier on example x is given by

$$d_i(x) = \begin{cases} 0 & \text{if } C_i(x) = C^*(x) \\ -x & \text{otherwise} \end{cases} \tag{4}$$

To compute the diversity of an ensemble of size n, on a training set of size m, we average the above term:

$$\frac{1}{nm} \sum_{i=1}^{n} \sum_{j=1}^{m} d_i(x_j) \tag{5}$$

This measure estimates the probability that a classifier in an ensemble will disagree with the prediction of the ensemble as a whole. Our approach is to build ensembles that are consistent with the training data and that attempt to maximize this diversity term.

In this section we introduced a new meta-learner algorithm that uses an existing learner to build an effective diverse committee in a simple, straightforward manner. In our algorithm (see Algorithm 5.2.1), an ensemble is generated iteratively, first learning a classifier and then adding it to the current ensemble. We initialize the ensemble to contain the classifier trained

on the given training data for the single database that contains the target relation. The classifiers in each successive iteration are trained on the original training data combined with some new training data, that is reachable by cross-database links that we add to the system in each step. We explain how these links are selected and added later. These cross-database links are approximate foreign-key links discovered in previous section. We refer to the labeled newly added training set as the *diversity data*. We train a new classifier on the union of the original training data and the diversity data, thereby forcing it to differ from the current ensemble. Therefore adding this classifier to the ensemble should increase its diversity. While forcing diversity we still want to maintain training accuracy. We do this by rejecting a new classifier if adding it to the existing ensemble decreases its training accuracy. This process is repeated until we reach the desired committee size or exceed the maximum number of iterations.

Ensemble Creation Algorithm:
Input:
BaseLearn – base learning algorithm
T: training examples;
C_{size}: desired ensemble size
I_{max}: maximum number of iterations to build an ensemble
R: ranked list of cross-database links
1. $i = 1$; *trials* = 1; C_i = *BaseLearn(T)*;
2. Initialize ensemble, $C^* = \{C_i\}$
3. Compute ensemble error, ε
4. While $i < C_{size}$ and *trials* $< I_{max}$
5. Select a link, $l \in R$, update *T* by adding new tuples reachable from *T* by *l*
6. $C' =$ *BaseLearn(T)*
7. $C^* = C^* \cup \{C'\}$
8. Compute training error, ε', of C^*
9. If $\varepsilon' < \varepsilon$
10. $i = i + 1$
11. $\varepsilon = \varepsilon'$
12. Otherwise,
13. $C^* = C^* - \{C'\}$
14. *trials* = *trials* + 1

To classify an unlabeled example, x, we employ the following method. Each base classifier, C_i, in the ensemble C^* provides probabilities for the class membership of x. If $\hat{P}_y(x)$ is the estimated probability of example x belonging to class y according to the classifier C_i, then we compute the class membership probabilities for the entire ensemble as:

$$\hat{P}_y(x) = \frac{\sum C_i \in C^* \hat{P}_{C_i,y}(x)}{|C^*|} \tag{6}$$

where $\hat{P}_y(x)$ is the probability of x belonging to class y. We then select the most probable class as the label for x, i.e. $C^*(x) = argmax_{y \in Y} \hat{O}_y(x)$

5.2.2 Link Selection

Various heuristics can be used in evaluating usefulness of a link when we want to choose and add a link to diversify the training data. To transfer information across heterogeneous databases, we must detect inter-database links, which are links between matched attributes and they can serve as bridges for information transfer. [29] proposes an approach for predicting the usefulness of links. They define the usefulness of a link as the maximum information gain of any feature generated by propagating information through this link. A regression-based approach for building a model to predict usefulness of links based on properties of links is proposed. However this approach is based on *Foil gain*, a variant of information gain, to measure the usefulness of a predicate and it assumes that our base learner is a rule based classifier. However we want our framework to be as general as possible and comply with any type of base learner. Hence we only use the confidence score returned by join discovery module to rank the cross-database links.

This link score is a weighted sum of two components: set resemblance score, and semantic similarity distance. In our implementation we treat both components equally. We rank all links and then in each step we pick one highest ranked link that has not been used previously. After all links have been tried for building a classifier, we start by adding two links in each step, and so on.

5.2.3 Why HeteroClass Should Work?

Ensembles of classifiers are often more accurate than their component classifiers if errors made by the ensemble members are uncorrelated. By training classifiers on oppositely labeled artificial examples, we reduce the correlation between ensemble members. Furthermore, the algorithm ensures that the training error of the ensemble is always less than or equal to the error of the base classifier; which usually results in a reduction of generalization error. Therefore, on average, using the predictions of an ensemble created in this way, we will improve the accuracy of the base classifier. Also we believe that diversity is the key to constructing good ensembles, and is thus the basis of our approach. Our approach diversifies the classifiers by extending the training data and including various data sources that capture different characteristics of data.

6 Empirical Evaluation

We now evaluate our HeteroClass framework in terms of its global accuracy. We also evaluate the effectiveness and sensitivity of each of the two major components

6.1 Experimental Settings

Data Sets: We use two real-world data sets DBLife and Inventory. The DBLife data set consists of six databases. Two of them are publication records extracted from DBLP database and their schema comply with the DBLP schemas used in Clio project [13]. Another database in DBLife that we are using is the "Researchers" database that contains information and

records on people (database related researchers), their associations with each other (e.g. co-authorship; co-working, etc.) and their metadata (e.g. department, school, location, etc.). The last DBLife database that we are using is the "DBWorld Events" database, that contains information about people, and events occurring in database community, example of which are giving a talk, or giving a service, etc. The classification task in this case was to predict the research topics of the researchers (which was the target relation). The Inventory data set consists of information about products, stores, availability of items, and associated inventory records on various items (e.g. books, CDs, etc.). We use five database in this dataset. The classification task in Inventory data set is to predict the availability of products in stores.

Experiment Methodology: In the experiments, we divide the databases in each dataset into two separate sets of equal size. We use one set for training and we use the third set for testing. The DBLife snapshot of the database contains information of 500 researchers. Each Inventory database contains about 1000 items. We limit the maximum size of the ensemble to 10.

6.2 Prediction Accuracy

In the first experiment, we evaluate the accuracy of the prediction of HeteroClass generated classifiers against two other algorithms. The first algorithm that we compare against is CrossMine [30], which is a single database multi-relational rule based classifier. The second algorithm is a variation of MDBM [29]. However in MDBM, the cross-database links are assumed to be known. This assumption does not hold in heterogeneous settings as of our experiments. So we modify this algorithm to use the links generated by our join discovery module. In this experiment, the iterative ensemble construction is repeated as long as the error in the training phase converges or we have added all the cross-database links to the training data set.

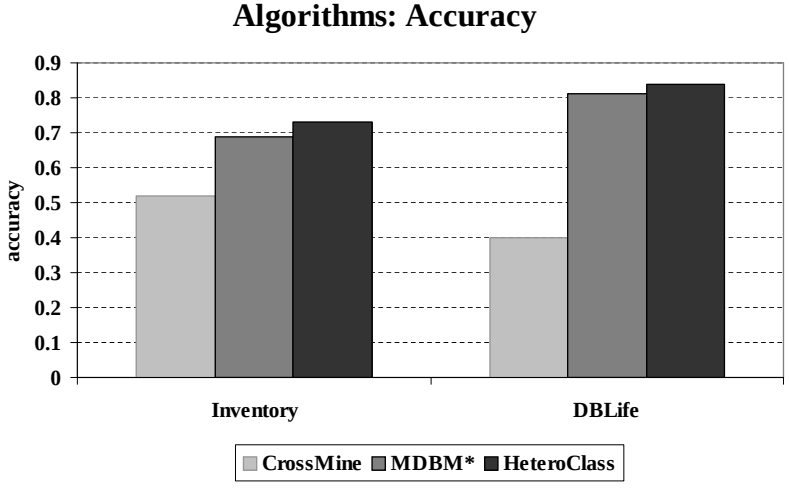

Figure 1: Comparing prediction accuracy of three algorithms in two domains.

Figure 1 compares the accuracy of the three algorithms. We can see that MDBM and

HeteroClass that are multi-database algorithms, achieve much higher accuracy compared to CrossMine which only uses the data available in a single database (that contains the target relation). The figure suggests that using an ensemble of classifiers improves the accuracy over the other multi-database classification algorithm in both domains.

6.3 Sensitivity Analysis

The previous experiment justifies the fact that HeteroClass improves the accuracy over other methods. However we are interested to evaluate various components of the system to find out how effective they are. In this section, we first evaluate our join discovery module, and then we study how sensitive the algorithm is to the size of the ensemble.

6.3.1 Join Discovery Accuracy

We now examine the accuracy of foreign-key joins that are suggested by the techniques of Section 5.1. To decide on accuracy of the join discovery procedure, we manually inspected each automatically identified foreign-key and marked it as correct or incorrect. We also compiled a list of all foreign-key joins that should have been automatically identified. With this information, we compute the precision of the join discovery procedure as the fraction of identified foreign-key joins that are correct; in turn, we compute the recall of this procedure as the fraction of the correct foreign-key joins that are identified. We combine precision and recall into a single metric, using the F_1-measure, defined as $F_1 = \frac{2PR}{P+R}$, where P stands for precision and R stands for recall.

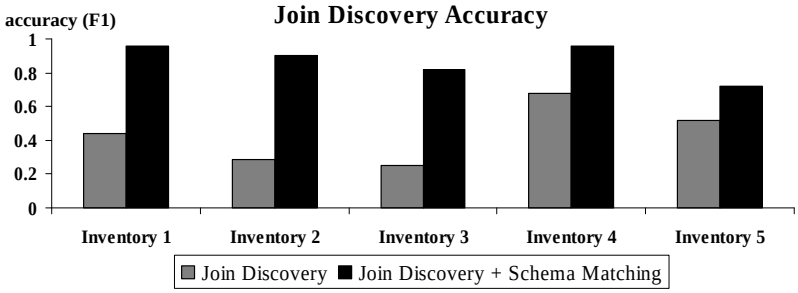

Figure 2: Effect of schema matching to join discovery algorithm.

Figure 2 reports the F_1-measure value for the join-discovery procedure defined in Section 5.1, which includes both structure discovery and schema matching step; this combined procedure, which we developed for the our HeteroClass, is denoted as "Join Discovery + Schema Matching" in Figure 2. We also report the F_1-measure value for a simpler, state-of-the-art join discovery procedure that does not rely on schema matching [5]; this procedure is labeled "Join Discovery" in Figure 2. Each pair of bars in Figure 2 corresponds to choosing one Inventory database and discovering foreign-key joins with four other Inventory databases. The results show that schema matching significantly improves the accuracy of the join discovery method, by 15-49%. This demonstrates the effectiveness of incorporating schema matching techniques into join discovery.

6.3.2 Effect of Ensemble Construction Algorithm

We now examine how effective our ensemble construction algorithm is. To do so, for each dataset, we repeat the first experiment, but limit the size of the ensemble.

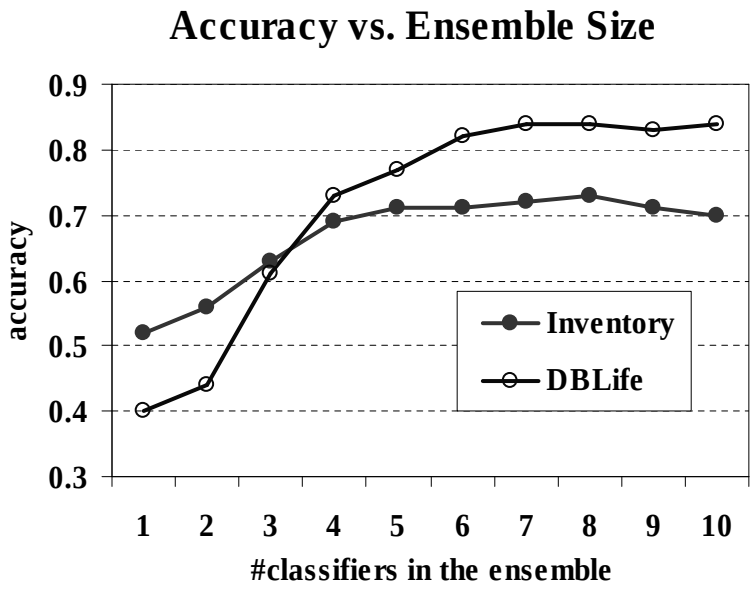

Figure 3: Effect of ensemble size to prediction accuracy.

Figure 3 depicts the sensitivity of accuracy to the size of the ensemble. We can see that as we add more links to the training data and as we increase the size of the ensemble, the prediction accuracy is increased.

7 Conclusion

In this project, we have studied the problem of classification from multiple heterogeneous databases. We addressed the main arising challenges in this context, namely, heterogeneity and attribute instability. We developed HeteroClass framework that addresses these challenges using best-effort data integration techniques and ensemble approach in building prediction models. We evaluated our methods using two real-world data sets and show the improvement over state-of-the-art applicable methods in this contexts.

References

[1] R. Ananthakrishna, S. Chaudhuri, and V. Ganti. Eliminating fuzzy duplicates in data warehouses. In *VLDB*, 2002.

[2] P. Andritsos, R. J. Miller, and P. Tsaparas. Information theoretic tools for mining database structure. In *SIGMOD*, 2004.

[3] D. W. Cheung, V. T. Ng, A. W. Fu, and Y. Fu. Efficient mining of association rules in distributed databases. In *TKDE*, 1996.

[4] W. Cohen. Integration of heterogeneous databases without common domains using queries based on textual similarity. In *SIGMOD*, 1998.

[5] T. Dasu, T. Johnson, S. Muthukrishnan, and V. Shkapenyuk. Mining database structure; or, how to build a data quality browser. In *SIGMOD*, 2002.

[6] R. Dhamankar, Y. Lee, A. Doan, A. Halevy, and P. Domingos. iMAP: Discovering complex mappings between database schemas. In *SIGMOD*, 2004.

[7] T. Dietterich. Machine learning research: Four current directions. *AI Magazine*, 18(4):97136, 1997.

[8] H. Do and E. Rahm. COMA: A system for flexible combination of schema matching approaches. In *VLDB*, 2002.

[9] A. Doan and A. Halevy. Semantic integration research in the database community: A brief survey. *AI Magazine*, 26(1):83–94, 2005.

[10] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an RDBMS for Web data integration. In *WWW*, 2003.

[11] J. Han and M. Kamber. *Data Mining: Concepts and Techniques*. Morgan Kaufmann, 2000.

[12] L. K. Hansen and P. Salamon. Neural network ensembles. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 12(10):993–1001, 1990.

[13] M. Hernandez, R. J. Miller, and L. M. Haas. Clio: A semi-automatic tool for schema mapping. In *SIGMOD-01*.

[14] M. Hernandez and S. Stolfo. The merge/purge problem for large databases. In *SIGMOD*, 1995.

[15] Y. Huhtala, J. Kärkkäinen, P. Porkka, and H. Toivonen. TANE: An efficient algorithm for discovering functional and approximate dependencies. In *The Computer Journal 42(2)*, 1999.

[16] J. Kang and J. Naughton. On schema matching with opaque column names and data values. In *SIGMOD*, 2003.

[17] M. Kantarcioglu and C. Clifton. Privacy-preserving distributed mining of association rules on horizontally partitioned data. In *TKDE*, 2004.

[18] L. Kuncheva and C. Whitaker. Measures of diversity in classifier ensembles and their relationship with ensemble accuracy. *Machine Learning*, 51(2):181207, 2003.

[19] Y. Lindell and B. Pinkas. Privacy preserving data mining. In *CRYPTO*, 2000.

[20] A. McCallum, K. Nigam, and L. Ungar. Efficient clustering of high-dimensional data sets with application to reference matching. In *SIGKDD*, 2000.

[21] S. Melnik, H. Molina-Garcia, and E. Rahm. Similarity Flooding: A Versatile Graph Matching Algorithm. In *ICDE*, 2002.

[22] P. Melville and R. J. Mooney. Constructing diverse classifier ensembles using artificial training examples. In *Proceedings of the Eighteenth International Joint Conference on Artificial Intelligence (IJCAI-2003)*, 2003.

[23] J. C. Pinheiro and D. X. Sun. Methods for linking and mining massive heterogeneous databases. In *KDD*, 1998.

[24] E. Rahm and P. Bernstein. On matching schemas automatically. In *VLDB Journal 10(4)*, 2001.

[25] S. Sarawagi and A. Bhamidipaty. Interactive deduplication using active learning. In *SIGKDD*, 2002.

[26] K. Tumer and J. Ghosh. Error correlation and error reduction in ensemble classifiers. *Connection Science*, 8(3-4):385–403, 1996.

[27] J. Vaidya and C. Clifton. Privacy preserving association rule mining in vertically partitioned data. In *KDD*, 2002.

[28] J. Vaidya and C. Clifton. Privacy-preserving k-means clustering over vertically partitioned data. In *KDD*, 2003.

[29] X. Yin and J. Han. Efficient classification from multiple heterogeneous databases. In *PKDD*, 2005.

[30] X. Yin, J. Han, J. Yang, and P. S. Yu. Crossmine: Efficient multi-relational classification. In *ICDE*, 2004.

[31] X. Yin, J. Han, and P. S. Yu. Cross-relational clustering with user's guidance. In *KDD*, 2005.

[32] G. Zenobi and P. Cunningham. Using diversity in preparing ensembles of classifiers based on different feature subsets to minimize generalization error. In *Proceedings of the European Conference on Machine Learning*, page 576587, 2001.

[33] N. Zhong, Y. Yao, and M. Ohshima. Peculiarity oriented multidatabase mining. In *IEEE Trans. on Knowledge and Data Engineering. 15 (4)*, 2003.