2011

Learning predictive models from massive, semantically disparate data

Neeraj Koul
Iowa State University

Learning predictive models from massive, semantically disparate data

by

Neeraj Koul

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Iowa State University

Ames, Iowa

2011

DEDICATION

This thesis is *dedicated* to my uncle and inspiration Dr. Gridhari Lal Koul, paternal grandfather Mr. Radha Krishen Koul and maternal grandfather Mr. Jagar Nath Razdan, all of whom were with me at the start of the PhD program but sadly not at the end. You all are greatly missed.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACKNOWLEDGEMENTS

When your graduate studies encompasses stints of two and four years spread over ten years, a marriage, two kids and three jobs it is no wonder that you have a lot of people to thank. First of all I want to thank my major advisor Dr. Vasant Honavar for keeping me motivated and working around a multitude of my constraints. Dr. Honavar, thanks for always being fair, considerate and available. I also want to thank Dr. Samik Basu for being a wonderful source of inspiration. I also want to thank the rest of my committee members, Dr. Shashi Gadi, Dr. Robyn Lutz and Dr. Simanta Mitra for all their help and useful advice. I want to acknowledge that this research was supported in part by the grant IIS 0711356 to Vasant Honavar from the National Science Foundation.

I want to thank Dr. Rattan Lal Hangloo for his constant encouragement during each and very phone call that I made to him. I want to thank Mr Rajesh Kumar Raina for always keeping me in his prayers. I want to thank my sister Sunita Koul for constantly reminded me to get it done and my brother in law Anil to teach me, by example, to be calm in all situations. I want to thank my siblings Komal, Sheetal, Ishan, Saiensh for looking up to me so much that I could not envision to fail. I want to thank my in laws, Mr. Kuldeep Kumar Bhat , Mrs. Vijay Bhat, Mr. Yugdeep Bhat and Mrs. Rashmi Bhat for providing their support and confidence . I want to thank Koul Sahib, Tathee, Veena Ma, Reeta Jee, Raju Baya, Baijan, Ravi Jee and Dolly Didi for everything. You people give meaning to my life and anything useful I do is a result of your prayers and blessings.

I want to thank Chander Mohan Raina for his continued friendship. I also want to thank the multitude of friends that I have made during my stay at Iowa State. These include Smruti, Mayuresh, Nikhil, Siju, Bhooshan, Renuka, Gondi, Rahul, Murali, Chinni, Rajee, Moti, Vidya, Shantha, Prem, Abhijeet , Vani, Vishwa, Anupreet and all the people involved with Sankalp. I also want to thank members of the AI lab for their insightful discussions and friendship. These include (among others) Adrian, Doina, Fadi, Yasser, Jia Tao, Jie Bao, Jyotishman Pathak, Cornelia, Oksana, Ganesh, Bhavesh, Harris, Li Xue, Bui and Rafael.

Finally, I want to thank my wife Pratibha Bhat for being a rock solid support. You had stated that you must not be in the acknowledgment if you do not deserve it. Well you deserve atleast half (possibly more) share of *our* PhD degree. Without your support I absolutely would not have been able to complete this work. And last but not the least, a big shout of thanks to my beautiful daughters Sheen and Shereen for their inspiring smiles and allowing me to work when I had to.

ABSTRACT

Machine learning approaches offer some of the most successful techniques for constructing predictive models from data. However, applying such techniques in practice requires overcoming several challenges: infeasibility of centralized access to the data because of the massive size of some of the data sets that often exceeds the size of memory available to the learner, distributed nature of data, access restrictions, data fragmentation, semantic disparities between the data sources, and data sources that evolve spatially or temporally (e.g. data streams and genomic data sources in which new data is being submitted continuously). Learning using statistical queries and semantic correspondences that present a unified view of disparate data sources to the learner offer a powerful general framework for addressing some of these challenges. Against this background, this thesis describes (1) approaches to deal with missing values in the statistical query based algorithms for building predictors (Nave Bayes and decision trees) and the techniques to minimize the number of required queries in such a setting. (2) Sufficient statistics based algorithms for constructing and updating sequence classifiers. (3) Reduction of several aspects of learning from semantically disparate data sources (such as (a) how errors in mappings affect the accuracy of the learned model and (b) how to choose an optimal mapping from among a set of alternative expert-supplied or automatically generated mappings) to the well-studied problems of domain adaptation and learning in presence of noise and (4) a software for learning predictive models from semantically disparate data.

CHAPTER 1. INTRODUCTION

1.1 Background and Motivation

Machine learning approaches offer some of the most successful techniques to learn from data [Bishop (2006), Mitchell (1997)]. However, recent advances in high throughput data acquisition technologies in many areas have led to a proliferation of a multitude of physically distributed, autonomous and often semantically disparate data sources presenting several significant challenges in learning from data in such a setting [Caragea et al. (2005), Honavar and Caragea (2008)]: (1) *massive data size* that often exceeds the size of memory available to the learner, (2) *distributed nature of data*, and (3) autonomous data sources that may place certain restrictions on the access to the data. In addition, the advent of *Cloud Computing* [Wang et al. (2010), Vouk (2008), Armbrust et al. (2010)] offers the possibility for autonomous organizations to provide data analysis services over the cloud [Wang et al. (2008), Stein (2010)]. This calls for algorithms that can build predictive models from datasets accessible over the cloud. Further, autonomous data sources, even in related domains, are often designed and developed independent of each other (e.g. Apweiler et al. (2004) lists multiple protein sequence databases). As a result, independently developed data sources differ in structure (relational databases, flat files) as well as the type of capabilities they provide a user to access the data (e.g. support for SQL queries, ability to execute user-supplied code, web services etc.). Further, the rise of the Semantic Web [Berners-Lee et al. (2001), Davies et al. (2002), Antoniou and van Harmelen (2008)] has resulted in a proliferation in the use of ontologies to associate semantics with data. Since ontological commitments associated with the data source are often determined at design time, semantic disparity is a natural outcome of independently developed and autonomously developed data sources. Hence, different data sources often use disparate vocabularies (e.g., M.S. student versus Masters student), units (e.g., temperature measured in degrees Centigrade versus Fahrenheit), and levels of detail (e.g. graduate student, student) to describe the objects of interest in the world being modeled. Learning a predictive model from such disparate data sources (say in some domain of interest to a learner) requires reconciliation of the semantic differences between the learner's conceptualization of the world (i.e. the learners ontology) and the models of the world associated with the disparate data sources (i.e., the data source ontologies). Furthermore, in the distributed setting, the learner must cope with *data fragmentation.* The fragmentation could be *horizontal* and/or *vertical* data fragmentation. In the case of horizontal fragmentation, each data source (e.g., economic data for different states) contains a subset of data tuples that make up the data source of interest (e.g., economic data for the nation). In the case of vertical fragmentation, the different data sources contain subtuples of data tuples that make up the data source of interest. In addition, in settings such as learning from genomic data where new data is being constantly submitted to the data sources, the learners need the ability to incorporate new data without having to rebuild the learned model from scratch. The aforementioned challenges show that there is an urgent need for approaches to learning predictive models from large datasets (that cannot fit in the memory available on the device where the learning algorithm is executed) that are scalable, are able to incorporate data updates, do not require access to the underlying dataset, are able to handle data fragmentation (horizontal as well as vertical) and are able to cope with semantic disparity (both at the schema level and the data content level).

Despite attention to several aspects of this problem in literature (see Section 1.1.1), there are significant gaps in the current state of the art that remain to be addressed. Specifically, theoretically well-founded and practical approaches to learn predictive models from data are needed in the setting where:

- access to the underlying data (which may contain missing values) is unavailable (due to massive size or access restrictions) but the data source provides certain statistics over the data.

- the data is fragmented (horizontally as well as vertically) from the learner's point of view.

- the mapping used to resolve semantic disparity contains errors.
- choosing an optimal mapping (to resolve semantic disparity) from among a set of available candidate mappings (either provided by different experts or generated using one of the several available ontology alignment methods).

This thesis attempts to address these challenges.

1.1.1 Related Work

Scaling learning algorithms to large data sets have received significant attention in literature. Provost and Kolluri (1999) and Grossman (2001) survey work on scaling up learning algorithms. Examples of approaches that have been explored include parallelization of specific algorithms [Jin and Agrawal (2003)], support for disk resident data [Alsabti et al. (1998), Zou et al. (2006)], and learning decision trees from statistical queries [Moore and Lee (1998), Caragea et al. (2004a), Bar-Or et al. (2005)]. Alternate approaches to learn from intractably large data sets include distributing the centralized data set to multiple processors [Chu et al. (2006)], using a tractable subset of the entire data set [Wilson and Martinez (2000), Molina et al. (2002), Czarnowski and Jedrzejowicz (2008)] or learning models on tractably sized subsets of the data and then combing the models as in ensemble of classifiers [Hall et al. (2000), Breiman (1999)]. Of related interest is *Apache Mahout* [Foundation (2010), Ingersoll (2009)], the goal of which is to build scalable machine learning algorithms. Mahout aims to exploit the power of a distributed computing platform that is based on the Map/Reduce programming paradigm [Dean and Ghemawat (2008)] to scale predictive models to large datasets.

The task of specifying the mappings has been extensively studied in the semantic web, database and the ontology communities in various contexts including *ontology mapping*, *ontology matching*, *ontology alignment*, *ontology merging* and *ontology integration* (see survey papers Choi et al. (2006), Doan and Halevy (2005), Kalfoglou and Schorlemmer (2005), Wache et al. (2001) and Noy (2009) and the book Euzenat and Shvaiko (2007)). The Ontology Alignment Evaluation Initiative [OAE (2010)] is a coordinated international initiative that is aimed at developing a set of common metrics and benchmarks for evaluating ontology alignment methods . A variety of techniques for automating the specification of mappings between ontologies have been explored. These include schema based approaches, instance based approaches or a combination of the two (see Euzenat and Shvaiko (2007) for a detailed discussion of these methods). Shvaiko and Euzenat (2008) identify ensuring consistency of mappings, specially in large ontologies, identified as one of the major challenges in ontology integration. Barring a few exceptions [Meilicke et al. (2007), Meilicke et al. (2009), Kalyanpur et al. (2006), Falconer et al. (2007)] the problem of eliminating inconsistent mappings has received limited attention in the literature.

Distributed data mining has received considerable attention in literature [see Park and Kargupta (2002)]. Domingos (1997), Prodromidis et al. (2000) and Aoun-Allah and Mineau (2007) propose *ensemble of classifiers* approach to learning from horizontally distributed fragmented data. The said approach involves learning classifiers on each subset of data and combining them in some well defined way (e.g. using a weighed voting scheme). Algorithms to learn decision trees from vertically fragmented data have been proposed by Bhatnagar and Srinivasan (1997). Most proposed approaches to learning from semantically disparate data sources involve the use of a middleware (such as a data integration system or a query answering engine) to resolve semantic heterogeneity (e.g. Chawathe et al. (1994), Caragea et al. (2005)).

The problem of data integration (see Lenzerini (2002) for theoretical overview) has received significant attention in literature (see Doan and Halevy (2005) for a survey). Data integration systems offers a uniform interface to multiple data sources and have enjoyed significant success both in research and commercial applications [Halevy et al. (2005), Halevy et al. (2006), Haas (2007)]. A significant portion of the work in this area has focused on bridging semantic differences between schema and ontologies associated with the individual data sources (see Shvaiko and Euzenat (2005),Wache et al. (2001) for surveys of approaches that address schema heterogeneity). The approach in SIRUP [Ziegler and Dittrich (2004), Ziegler and Dittrich (2007)]] primarily addresses schema heterogeneity. Aspects of data content heterogeneity are addressed in Wache and Stuckenschmidt (2001), Goh et al. (1999). The handling of data content heterogeneity in COIN [Goh et al. (1999)] is limited to unit conversions (e.g., dollars into euros) and term substitutions. BUSTER [Visser et al. (2000)] handles both schema and data heterogeneity but assumes an existence of a global ontology with each data source ontology being a refinement of the global ontology. Haas et al. (2009) propose a framework that leverages information about both schema and data to improve the integrated result. Cafarella et al. (2009) describes Octopus, a system that combines search, extraction, data cleaning and integration, and enables users to create new data sets from those found on the Web. Some recent work has also focused on data integration with uncertainty [Agrawal et al. (2010), Dong et al. (2007)] where uncertainty may be in the data itself, in terms of the mappings used to resolve the semantic heterogeneity, errors in the data or the queries posed to the data (say posed using keywords rather than in the structured form).

The current approaches are either unable to handle massive data sets because they need in memory access to the data as is in the case of WEKA [Witten and Frank (2005)] and other similar tools, or assumes direct access to data to handle missing values in data (e.g. WekaDB [Zou et al. (2006)]). However, in many practical applications, the large size, access restrictions, memory and bandwidth constraints, and in some instances, privacy considerations prohibit direct access to data. *Ensemble* based approaches [Dietterich (2000a)] to learn from distributed data may require user specified code to be executed on the multiple data sites and do not provide rigorous guarantees with respect to the centralized counterpart. Mahout [Ingersoll (2009)] and Hadoop based systems [Ghoting and Pednault (2009), Chu et al. (2006)] that divide a massive dataset among multiple clusters and provide guarantees with respect to centralized learning are unable to learn in the semantically disparate setting. Approaches to learn in the semantically disparate setting (e.g. Caragea et al. (2005)) are unable to cope with both horizontal and vertical fragmentation. In addition, though there has been some attention given to the task of detecting mapping errors [Meilicke et al. (2007), Meilicke et al. (2009), Kalyanpur et al. (2006), Falconer et al. (2007)], there has been very little work on the task of learning in presence of errors. Hence, there is a need for approaches to learn from semantically disparate data sources that are fragmented from a learner's point of view and may deny access to the underlying data. In addition, in such a setting, the learning algorithms used must be robust in presence of errors in the mappings.

1.2 Our Approach

This thesis addresses learning of predictive models from *semantically disparate data* in the following setting:

- the data is distributed.
- bandwidth limitations, constant updates or privacy concerns prevent the data being transported to a centralized location.
- data is fragmented both horizontally and vertically from a learner's point of view.
- the database schema used by the various autonomous data sources differ from the learner's schema and from each other.
- the ontologies used by the autonomous data source to associate meaning with data are related but differ from each other and the ontology used by the learner.
- mappings are used to resolve both the schema heterogeneity and data content heterogeneity.

Our approach to address this problem, building on the work of Caragea et al. (2005), is to learn a predictive model from a dataset by interacting with the data source(s) (that holds the dataset of interest to the learner)) only through statistical queries. This approach allows us to cope with the challenges of massive data size (since in general the statistics of the data are much smaller than the size of the data), no access to underlying dataset (because it interacts with data source only through statistical queries) and in certain cases, data source updates (additions, deletions of large subsets of data). In addition, we introduce a data integration framework that presents to the learner a collection of physically distributed, autonomous and disparate data source as though they were a single data source structured according to an ontology supplied by the user. The use of a data integration framework allows us to simultaneously cope with data fragmentation and semantic disparity when learning in the setting of disparate data sources. In addition, we reduce some important problems that occur in the setting of learning from disparate data (e.g. learning in presence of mapping errors, choosing among multiple available mappings) to some well-studied problems in literature such as domain adaptation and learning in presence of noise. This reduction opens up the possibility of applying techniques and algorithms from these settings to the setting of learning from disparate data.

1.3 Thesis Outline

The rest of the dissertation is organized as follows. In what follows, we summarize the rest of the chapters in the thesis. Each chapter is self contained and corresponds to a paper. Some of the work presented in this thesis has been already published in conference proceedings, is under review, or is in preparation.

Chapter 2 describes an open source framework (Indus Learning Framework) that learns Naive Bayes and decision trees classifiers from autonomous data sources using count queries. The framework aims to address the challenge of learning from data when access to the underlying data is unavailable (say due to constraints such as privacy or the cost of moving the massive data set to a centralized location). In this setting we describe techniques to apply data pre-processing steps (such as data filtering and handling missing values) by incorporating their effect on the statistics required to build the classifiers. In addition, we present some optimization techniques that can be used to minimize the number of queries required to build a classifier from the dataset.

Chapter 3 describes an open source data integration system (Indus Integration Framework) that presents the semantically disparate data sources that are fragmented from a learner's point of view as a single data source to a learner. The heart of the system is an query planner that answers a user posed query in terms of queries that can be answered by the semantically disparate data sources, using available schema mappings, inter ontology mappings and conversion functions.

Chapter 4 presents an approach to learn from massive sequence data using only count queries. Specifically, in this chapter we focus attention on *Markov Property* based class of predictive models for sequences: Markov Models, Probabilistic Suffix Trees, Interpolated Markov Models that are among some of the most widely used in sequence classification.

Chapter 5 describes the problem of learning in the setting where the mappings used to resolve semantic disparity between the multiple data sources may contain errors. Specifically, in this chapter we show that the problem of learning from semantically disparate data sources in the presence of mapping errors can be reduced to a variant of the problem of learning in the presence of *nasty* classification noise (a problem previously studied in literature). This reduction allows us to transfer theoretical results and algorithms from the latter to the former.

Chapter 6 describes, *Learning Scenario*, a model for supervised learning that can be used to model distributed learning, learning from disparate data sources and domain adaptation (among others). We use *Learning Scenario* to explore connections between domain adaptation and learning from disparate data. Specifically we show that choosing among multiple available mappings can be reduced to an aspect of domain adaptation and any two domains can be adapted using probabilistic mappings.

Chapter 7 concludes the thesis, and provides a summary of contributions and directions for further research.

CHAPTER 2. Learning Classifiers from Large Databases Using Statistical Queries

The emergence of data rich domains has led to an exponential growth in the size and number of data repositories, offering exciting opportunities to learn from the data using machine learning algorithms. In many applications, the learning algorithm may not have direct access to the entire dataset because of massive size of data , access restrictions, or bandwidth requirements. In such settings, there is a need for techniques that can learn predictive models (e.g., classifiers) from large data sets without direct access to the data. We describe an approach to learn from large databases using statistical queries. Specifically, we extend previous approaches to learning from statistical queries to handle missing values in data in settings where direct access to the data or the ability to execute user-supplied procedures on the data repositories is prevented. We demonstrate the proposed approach by providing an open source implementation of Naive Bayes and Decision Trees learning algorithms in a setting where the data repositories are large relational databases that only answer SQL count queries. We analyze the *query complexity* (a measure of the number of queries needed) for constructing classifiers in such settings.

2.1 Introduction

Advances in virtually every area of human endeavor are being increasingly driven by our ability to acquire knowledge from vast amounts of data. Most current approaches to learning from data assume direct access to data. However, in many practical applications, the large size, access restrictions, memory and bandwidth constraints, and in some instances, privacy considerations prohibit direct access to data. Hence, there is an urgent need for scalable approach to learning predictive models from large datasets (that cannot fit in the memory available on the device where the learning algorithm is executed). To address this need, especially in settings where the data reside in distributed repositories, Caragea et al. [Caragea et al. (2004a), Caragea et al. (2005)] have introduced a general strategy for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the remote data sources. This involves separating a learning algorithm into two components: (i) a statistical query [1] generation component that poses a set of statistical queries to be answered by a data source and (ii) a hypothesis construction component that uses the resulting statistics to modify a partially constructed hypothesis (and may further invoke the statistical query component as needed). The implementation of this strategy in practice requires effective methods for minimizing the cost of statistical queries, and for coping with missing values in data. This chapter describes an approach to learning predictive models (e.g., decision trees) from large databases using statistical queries that is guaranteed to yield the same results as those obtained by the corresponding learning algorithms when they have direct access to data.

2.2 Learning Predictive Model From Semantically Disparate Data Sources

Our approach to learn in a setting where access to underlying data is unavailable is based on the observation that for a number of learning algorithms, relevant statistics of the data are adequate to build predictive models. In particular, Caragea et al. [Caragea (2004); Caragea et al. (2004a)] introduced a general strategy for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries. In the case of decision trees and Naive Bayes classifiers , statistical queries generated by the learning algorithms take the form of queries for counts of data instances that match specific constraints on the values of certain attributes and class labels. Such counts are easy to obtain from a relational database or other types of

[1]A statistic is simply a function of a dataset; A statistical query returns a statistic (e.g., the number of instances in the dataset that have a specified value for a specified attribute.)

data repositories that support such count queries. Hence, these predictive models can be built if the data source provide the relevant counts(say via queries) even if it denies access to the entire data set. In addition, this approach also addresses the problem of the inability of the learner to load the entire data set in memory (since, in general the size of statistics required is much smaller than size of the entire dataset). Based on these observations we present, Indus Learning Framework (ILF), a system that learns from data using only statistical queries. The framework assumes no access to underlying data and does not depend on execution of user defined functions or stored procedures on the data sources. It handles missing values and application of filters by dealing with their equivalent effect on the statistics obtained over the data and as such does not need access to the underlying data set. Subsequently, we extend the ILF with an integration framework (called Indus Integration Framework) that extends the capability of ILF to learn from semantically disparate data sources that are fragmented from the learner's point of view.

2.2.1 Indus Learning Framework

In the Indus Learning Framework, we assume that each data source D has a data descriptor $Desc(D)$ that describes the structure of the data (attributes and their domains) over which the predictive model is to be built. Formally $Desc(D) =< A, C, \mathcal{V} >$ where $A = \{a_1, a_2, \ldots, a_n\}$ is the set of attributes, $C \notin A$ a special attribute corresponding to the class label, $\mathcal{V} = \{V_{a_1}, \cdots V_{a_n}, V_C\}$ a set of *domains* where $V_{a_i} = \{v^i_1 \cdots v^i_{m_i}\}$ is the set of possible values of attribute a_i ($m_i = |V_{a_i}|$) and V_C the set of possible class labels. To keep things simple, we assume that all the attributes are nominal. Given $Desc(D)$ the instance space $I = V_{a_1} \times V_{a_2} \times V_{a_3} \cdots V_{a_n}$. We assume that the data source is a relational database and $Desc(D)$ implicitly specifies the schema of the database as follows: the dataset is stored in a table named D and it has columns $\{a_1, a_2, \ldots, a_n\}$ corresponding to the attributes in A, and the column C corresponds to the class label. A dataset D is a multiset whose elements belong to $I \times V_C$. $Desc(\mathsf{D})$ is used to formulate the queries that are posed against the dataset D. Suppose the data source D supports a set of primitive queries Q_D. We assume the primitive queries in Q_D

are posed in κ, the query language supported by the data source. In our setting the primitive queries correspond to count queries against D. When D is a relational database, κ is *SQL* and the count queries take the form: *Select Count(*) from D where* $C = c_k$ *AND* $a_i = v_j^i$ represented as $S(D, C = c_k, a_i = v_j^i)$. Similarly the count query *Select Count(*) from D* is represented as $S(D)$.

We assume that the learner expresses statistical queries against D in its own statistical query language Λ. The ILF includes a *query planner* Π that transforms a query $q(s_D)$ expressed in Λ for a statistic s_D into a plan for answering s_D using some subset of the primitive statistical queries Q_D. We assume that the query planner Π has at its disposal, a set of operators $\oplus$ that can be used to combine the answers to queries in Q_D to obtain a statistic s_D. In the case where Q_D correspond to count queries, $\oplus$ may include $+, -,$.

Definition 1 *A query plan for a statistic s_D is an expression tree in which each leaf node corresponds to a* primitive query *in Q_D and each non-leaf nodes corresponds to an operator in $\oplus$ such that the evaluation of the expression tree returns the value of the statistic s_D*

We assume that the planner Π is guaranteed to produce a correct plan for every statistic s_D that is expressible in Λ. In general, there might be multiple query plans that can produce a given statistic s_D. For example, the plans $Select\ Count(*)\ from\ D\ where\ C = c_k$ and $\sum_{v_j^i \in V_i} S(D, C = c_k, a_i = v_j^i)$ yields the same statistic. While the first of these two plans may seem like the obvious one to choose, if all answers to the primitive queries used by the second plan are available to the system (perhaps because of other queries that have been executed and the results cached), it might be preferable to simply reuse the available results by choosing the second plan. We denote by $Q(p(s_D)$ the primitive queries used by the plan p for statistic s_D.

Let $S^L(D)$ be the set of statistics required by learning model L to build a predictive model over L. In general the calculation of $S^L(D)$ can be divided into n steps where $S_i^L(D)$ is the statistics in the i^{th} step and $\forall 1 > i > n$, $S_{i+1}^L(D)$ cannot be calculated before step $S_i^L(D)$. However, the learning algorithm L, when executed against a dataset D, generates a series of steps in order with the i^{th} step generating a *set* of statistical queries $S_i^L(D) = \{s_D(i, 1) \cdots s_D(i, n_i)\}$ where each query in S_i^L is expressed in Λ.

Definition 2 *A Query Plan P_1 for the statistics generated by L at the i^{th} step, $S_i^L(D) = \{s_D(i,1) \cdots s_D^L(i,n_i)\}$, is the set of query plans represented by $P_1(S_i^L(D)) = \{P_{1,1}(s_D(i,1)), \cdots P_{1,n_i}(S_D(i,n_i))\}$ where $P_{1,j}(s_D(i,j))$ is a* query plan *to compute the statistic $s_D(i,j)$.*

Definition 3 *A Query Plan P for $S^L(D)$, the statistics generated by L, is the set $P(S^L(D)) = \{P_1(S_1^L(D), \ldots P_n(S_n^L(D)\}$ where $P_i(S_i^L(D))$ is the Query Plan for the set of statistics generated by L in the i^{th} step.*

It follows that $Q(P_1(S_i^L(D))) = \cup_{j=1}^{n_i} Q(P_1, j(s_D(i,j)))$ denotes the subset of *primitive* queries against D that are required by a query plan P_1 for the i^{th} step of L. Similarly $Q(P(S^L(D)) = \cup_{i=1}^{n} Q(P_i(S_i^L(D))$ denotes the subset of *primitive* queries against D that are required by a plan P for $S^L(D)$. We define the *query complexity* of a plan $P(S^L(D))$, denoted by $QC(P(S^L(D)))$, as $|Q(P(S^L(D)))|$. The task of the query planner is to generate a plan $P(S^L(D))$ so as to minimize the query complexity $QC(P(S^L(D)))$ which may be important in settings where the data source imposes a cost for answering each primitive query. In addition to take advantage of the cache as outlined above, at each step i, the set of plans $P_i(S_i^L(D))$ can be optimized by sharing primitive queries across the query plans for individual statistical queries in the query set $S_i^L(D)$. We now introduce a few representative algorithms for learning from data sources in the ILF.

2.2.1.1 Naive Bayes Learner

Naive Bayes [Mitchell (1997)] is a simple learning algorithm that often yields classifiers with satisfactory performance in many applications. Naive Bayes classifier assigns an instance $\mathbf{x} =< x_1 \cdots x_n >$ to the most probable class label under the assumption that the attributes of the instance are independent given the class:

$$C_{NB}(\mathbf{x}) = arg\, max_{c_k \in V_C}\, P(c_k) \prod_{j=1}^{n} (P(x_i)|c_k)$$

During the learning phase, we need to estimate the class probabilities:

$$P(c_k) = \frac{S(D, C = c_k)}{S(D)}$$

and the probability of each possible value of each possible attribute for each class. That is probabilities of the form:

$$P(a_i = v^i_j | C = c_k) = \frac{S(D, C = c_k, a_i = v^i_j)}{S(D, C = c_k)}$$

where V_{a_i} denotes the domain of attribute a_i and $v^i_j \in V_{a_i}$ denotes the jth possible value in the domain V_{a_i} of the attribute a_i. Because learning Naive Bayes classifier requires, in the setting where the learner has direct access to the dataset, only a single pass through the dataset, in our setting, the learner needs to pose only a single set of queries against D, that is, a set of all queries of form $S(D), S(D, c_k)$ and $S(D, c_k, a_i = v^i_j)$. A Plan $P_{defualt}$ which poses all the possible queries of form $S(D), S(D, c_k)$ and $S(D, c_k, a_i = v^i_j)$ over *Desc(D)* has $QC(P_{default}) = 1 + |V_c| + \sum_{i=1}^{|A|} |V_c| \times |V_{a_i}|$.

2.2.2 Decision Tree Learner

Decision Tree algorithms [Quinlan (1993), Clark and Niblett (1989), Breiman (1984)] are among some of the most widely used machine learning algorithms for building classifiers from data. A decision tree learner recursively chooses at each step an attribute that yields the most information regarding the class label (e.g., as measured by the reduction in entropy). The choice of an attribute at a node in the decision tree partitions the dataset based on the values of the chosen attribute. This process is repeated until a desired termination criterion is satisfied. Hence, each path π from the root to a given node in the decision tree has associated with it, a subset of the data D^π. Extending the path π requires identifying an attribute that provides the maximal information about the class membership of instances in D^π. Given a dataset D^π the information gain for an attribute a_i, denoted by $Gain(D^\pi, a_i)$ is given by $H(D^\pi) - \sum_{a^i_j \in V_{a_i}} H(D^\pi_{a^i_j}) \times \frac{|D^\pi_{a^i_j}|}{|D^\pi|}$ where $D^\pi_{a^i_j}$ represents the sub data set of D^π where the attribute a_i takes the jth value (v^i_j) in its domain V_{a_i}. $H(D^\pi)$ denotes the *entropy* of the class distribution in D^πQuinlan (1993). We need to compute $Gain(D^\pi, a_i)$ using statistical queries against D. Let $S(D^\pi_{a_{ij}}, C = c_k, a_i = v^i_j)$ represent the count query over the dataset $D^\pi_{a^i_j}$ where the class label is c_k and the attribute a_i takes the value v^i_j. We have:

$$\begin{aligned} H(D^{\pi}_{a^i_j}) &= -\sum_{c_k \in V_c} \frac{S(D^{\pi}, C = c_k, a_i = v^i_j)}{|D^{\pi}_{a^i_j}|} \\ & \qquad log_2 \frac{S(D^{\pi}, C = c_k, a_i = v^i_j)}{|D^{\pi}_{a^i_j}|} \end{aligned}$$

$$H(D^{\pi}) = -\sum_{c_k \in V_c} \frac{S(D^{\pi}, C = c_k)}{|D^{\pi}|} log_2 \frac{S(D^{\pi}, C = c_k)}{|D^{\pi}|}$$

So queries of the form $S(D^{\pi}, c_k, a_i = v^j_i), S(D^{\pi}, c_k)$ and ones corresponding to $|D^{\pi}|$ i.e. $S(D^{\pi})$ and $|D^{\pi}_{a^j_i}|$ i.e. $S(D^{\pi}, a_i = a^j_i)$ constructed over $Desc(D^{\pi})$ allow us to determine the next node to be added to the decision tree. However, because $Desc(D^{\pi})$ is not available to the system, we need to compute it from $Desc(D)$ and the path π. Initially, the path π is empty, and the corresponding $Desc(D^{\phi}) = Desc(D) = \langle A, C, \mathcal{V} \rangle$. Consider a path π which is a one-step extension of a path ψ and obtained by appending an arc $a_i = a^j_i$ to ψ. Then $Desc^{\pi} = \langle A_{\pi}, C, \mathcal{V}_{\pi} \rangle$ where $A_{\pi} = A_{\psi} - \{a_i\}$ and $\mathcal{V}_{\pi} = \mathcal{V}_{\psi} - \{V_i\}$. For a path (π) in the construction of the decision tree that corresponds to attribute $a^{\pi}_1, a^{\pi}_2 \ldots a^{\pi}_{m_{\pi}}$ taking the values $v(a^{\pi}_1), v(a^{\pi}_2) \ldots v(a^{\pi}_{m_{\pi}})$ respectively, let $Clause(\pi)$ be a SQL fragment of the form $a^{\pi}_1 = v(a^{\pi}_1)\ AND\ a^{\pi}_2 = v(a^{\pi}_2)\ AND\ \ldots a^{\pi}_{m_{\pi}} = v(a^{\pi}_{m_{\pi}})$. It is easy to see that a query q over D^{π} can be expressed in terms of a query over D by simply adding to q, a clause $Clause(\pi)$. For example, $S(D^{\pi}, C = c_k, a_j = v^j_l)$ as *Select Count(*) From D Where* $C = c_k$ *AND* $a_j = v^j_l$ *AND* $Clause(\pi)$. Note the resulting query is a query against the data set D.

Consider a plan $P_{default}$ in which the step associated with dataset D^{π} poses all the queries of the form $S(D^{\pi}, c_k, a_i = v^j_i)$, $S(D^{\pi}, c_k)$, $S(D^{\pi})$ and $S(D^{\pi}, a_i = a^j_i)$. The query complexity $QC(P_{default})$ depends of the structure of the resultant decision tree $T(D)$. Let $\Omega(T(D)) = \{\pi_1, \pi_2 \ldots \pi_p\}$ be the paths (including empty path) in the resulting tree $T(D)$ and $l(\pi)$ be the length of the path π in $T(D)$ with the length of the empty path being zero. The Table 1.1 specifies the number of queries various type that are required for finding the attribute with the most information gain for D^{π}. We assume the leaf nodes also need to pose the same type of queries. Then summation over the types of queries in Table 1 for all the paths in $\Omega(T(D)))$ is $QC(P_{default}) = (1 + |V_C|) \times (|\Omega(T(D))| + \sum_{\pi \in \Omega(T(D))} \sum_{V_{a_i} \in \mathcal{V}_{\pi} - \{V_C\}} |V_{a_i}|)$. If $|V_{a_i}| = m$ for all attributes in A, then $QC(P_{default}) = (1 + |V_C|)(|\Omega(T(D))| + \sum_{\pi \in \Omega(T(D))}(|A| - l(\pi)))m$.

Table 2.1 Number of queries by $P_{default}$ for D^π.

Query Type	Number of Queries
$S(D^\pi)$	1
$S(D^\pi, C = c_k)$	$\|V_C\|$
$S(D^\pi, a_i = a_i^j)$	$\sum_{V_{a_i} \in \mathcal{V}_\pi - \{V_C\}} \|V_{a_i}\|$
$S(D^\pi, C = c_k, a_i = a_i^j)$	$\sum_{V_{a_i} \in \mathcal{V}_\pi - \{V_C\}} \|V_C\|\|V_{a_i}\|$

Decision Stump Learner: A *decision stump* [Ai and Langley (1992)] is a decision tree with a single node that is often used as a base learner in many ensemble approaches to learning [Kotsiantis et al. (2006), Martínez-Muñoz et al. (2007), Polikar (2006)]. It is straightforward to see that the queries required to build a decision stump are those required to split the root node in a decision tree and correspond to queries of the form $S(D)$, $S(D, C = c_k)$ and $S(D, C = c_k, a_i = a_i^j)$. Since these set of queries are the same as required for a building a Naive Bayes classifier, the plan $P_{default}$ for Naive Bayes (presented earlier) can be used to build a decision stump.

We now proceed to describe the application of filters in ILF.

2.2.3 Application of Filters

The ability to apply filters is an important step in the preprocessing of data and may have a critical bearing on the performance of the learner. In ILF we define the following types of filters:

- *Attribute Filter*: This filter, denoted by $F_{a_{remove}}$, removes the attribute a_{remove} from the data set D. The application of this filter is equivalent to updating $Desc(D)$ so that the value a_{remove} is removed from A. The application of this filter does not change the number of instances in D.

- *Class Value Filter*: This filter, denoted by $F_{c_{remove}}$, removes all the instances from the data set in which the class label takes the value c_{remove}. The application of this filter is equivalent to updating $Desc(D)$ such that value c_{remove} is removed from V_C and and modifying $S(D)$ as *Select Count(*) From D where* $C! = c_{remove}$. The modification to $S(D)$ is required since the application of this filter changes the number of instances in D.

- *Attribute Value Filter*: This filter, denoted by $F_{a_r=a_r^{remove}}$, removes those instances in the dataset D where the attribute a_r takes the value a_r^{remove} in its domain. The application of this filter is equivalent to updating $Desc(D)$ such that value a_r^{remove} is removed from V_{a_r}, modifying the query $S(D)$ as *Select Count(*) From D where* $a_r! = a_r^{remove}$ and modifying the query $S(D, c_k, a_i = a_i^j)$ as *Select Count(*) from D where* $C = c_k$ *AND* $a_i = a_i^j$ *AND* $a_r! = a_r^{remove}$. The modification to $S(D, c_k, a_i = a_i^j)$ is necessary to prevent counting the instances removed on the basis of $F_{a_i=a_r^{remove}}$ when getting the counts for an attribute a_i that is different from a_r.

- *Missing Value Instance Filter.* This filter, denoted by $F_?$, removes all instances from the dataset which have a missing value (indicated by marker ?). Given the set of attributes is $A = \{a_1, a_2 \ldots a_n\}$ the application of this filter involves appending of a $clause(\epsilon)$ to all the queries submitted where ϵ corresponds to the SQL fragment $a_1! =?$ AND $a_2! = ? \ldots a_n! =?$.

Note that application of the above filters in a decision tree corresponds to replacing D with D_π in the formulas outlined above (for each subdataset D_π associated with the π in the decision tree).

2.2.4 Dealing with Missing Values

The presence of missing values for some of the attributes in some of the instances in the dataset D requires modifications to the basic procedures described above for learning in ILF using statistical queries. The techniques for dealing with missing values that have been well studied in the literature [Liu et al. (1997), Quinlan (1993)] assume direct access to the dataset

D. In contrast, in our setting, we assume that access to the underlying dataset is unavailable. However, we assume that the database uses a designated marker (e.g., ?) to indicate a missing value and that the missing values occur only in attributes and not in the class label. The first approach to handling missing values is to remove from the dataset D all the instances that have a missing value. This approach, in the our setting , corresponds to applying the *Missing Value Instance Filter* $F_?$ described in section 2.2.3. Another approach to missing value is to treat the missing value marker (?) as an another possible value for the attributes. This approach corresponding to changing the $Desc(D)$ by updating V_{a_i} to $V_{a_i} \cup \{?\}$ for all a_i. However, this approach increases the query complexity since $|V_{a_i}|$ increases by one (recall the query complexity is dependent on $|V_{a_i}|$).

Another approach to handle missing values is to replace the missing value for an attribute with the value that occurs most frequently for the attribute in the dataset over which the predictive model is to be built . In the rest of the thesis we refer to this approach as *mode Imputation*. For, each $a_i \in A$ let $mode(a_i, D))$ return the value that occurs most frequently in D (in case of more than one possible value, the function returns one of the possible values and the choice may be based on some user preference). Since number of instance where an attribute a_i takes value a_i^j in dataset D is $S(D, a_i = a_i^j) = \sum_{c_k \in V_C} S(D, C = c_k, a_i = a_i^j)$, the calculation of $mode(a_i, D))$ in plan $P_{default}$ (for Naive Bayes) does no entail any additional queries. The imputation with the most likely value in Naive Bayes corresponding to updating the counts of the form $S(D, C = c_k, a_i = mode(a_i, D))$ to $S(D, C = c_k, a_i = mode(a_i, D)) + S(D, C = c_k, a_i =?)$. Since this requires additional queries of the form $S(D, C = c_k, a_i =?)$, it increases the query complexity of $P_{default}$ for Naive Bayes by $|A| \times |V_C|$.

Another imputation approach to handling missing values involves replacing the missing values probabilistically. In this technique the probability that a missing value for attribute a_i in D is a_i^j is the proportion in which the value a_i^j occurs for a_i in the dataset D (denoted by $P(a_i = a_i^j|D)$). We refer to this approach as *probabilistic imputation*. For Naive Bayes the *probabilistic imputation*, in our setting , corresponds to updating the counts of the form $S(D, C = c_k, a_i = a_i^j$ to $S(D, C = c_k, a_i = a_i^j) + S(D, C = c_k, a_i =?) \times P(a_i = a_i^j|D)$ where

$P(a_i = a_i^j|D) = \frac{\sum_{c_k \in V_C} S(D, C=c_k, a_i=a_i^j)}{\sum_{a_i \in A} \sum_{a_i^j \in V_{a_i}} \sum_{c_k \in V_C} S(D, C=c_k, a_i=a_i^j)}$. However, as before, queries of form $S(D, C = c_k, a_i = ?)$ are required and it increases the query complexity of $P_{default}$ for Naive Bayes by $|A| \times |V_C|$ (see optimization section on how to reduce the query complexity in presence of missing values).

For a decision tree handling missing values by mode imputation, in our setting, requires modification of $clause(\pi)$ associated with the subdataset $D(\pi)$. Let path (π) in the construction of the decision tree corresponds to attribute $a_1^\pi, a_2^\pi \ldots a_{m_\pi}^\pi$ taking the values $v(a_1^\pi), v(a_2^\pi) \ldots v(a_{m_\pi}^\pi)$ respectively. For each case where the value $v(a_i^\pi)$ is $mode(a_i^\pi, D)$ we modify the corresponding SQL fragment in $clause(\pi)$ from $a_i^\pi = v(a_i^\pi)$ to $a_i^\pi \ IN \ \{?, v(a_i^\pi)\}$. Such an approach has the same query complexity as $P_{default}$ in absence of missing values. Recall handling missing values by *probabilistic* imputation, involves the following procedure: during the construction of the decision tree when the attribute on which to perform the split (i.e. having the most information gain) has a missing value, it is probabilistically assigned to each subset based on the proportion of the possible values for the attributes. In the case when access to the underlying data is denied, this process requires the counts for each query posed over D^π to be modified appropriately to account for the probabilistic assignment of the missing value. Let $A(\pi)$ be the set of attributes that are part of π and $l(\pi) = |A(\pi)|$ be the length of the path π. For an instance that is part of dataset D^π, the attribute $a_i^\pi \in A(\pi)$ could either have one of the two possible values (i.e. $v(a_i^\pi)$ or ?). Each instance will contribute a different amount to the count based on whether an attribute $a_i^\pi \in A(\pi)$ has a missing value or $v(a_i^\pi)$ for that instance. For a path π there are $2^{l(\pi)}$ possible combination of $clause(\pi)$ (since there $l(\pi)$ attributes and each attribute take can take one of two possible values). Let $clause(\pi_r), 1 \leq r \leq 2^{l(\pi)}$ represent the different possible combinations of the attributes in π taking the missing value or the actual value in the path. Let $A(\pi_r, ?)$ be the attributes that take the value ? in $clause(\pi_r)$. Then $S(D^\pi, C = c_k, a_i = a_i^j) = \sum_{r=1}^{2^{l(\pi)}} w_r \times S(D, C = c_k, a_i = a_i^j) \ clause(\pi_r) \ where$

$$w_r = \begin{cases} 1 & when \ A(\pi_r, ?) = \phi \\ \prod_{a_i \in A(\pi_r, ?)} P(a_i = v^\pi(a_i)|D) & \text{otherwise} \end{cases} \tag{2.1}$$

and $P(a_i = v^{\pi}(a_i)|D)$ is the proportion in which the attribute a_i takes the value $v^{\pi}(a_i)$ in D and is calculated as before. Similarly, $S(D^{\pi}) = \sum_{r=1}^{2^{m_{\pi}}} w_r \times S(D)clause(\pi_r)$ and $S(D^{\pi}, a_i = a_i^j) = \sum_{r=1}^{2^{m_{\pi}}} w_r \times S(D, a_i = a_i^j)clause(\pi_r)$. The technique as described, to calculate a single count of the form $S(D^{\pi}, C = c_k, a_i = a_i^j)$ or $S(D)$ or $S(D^{\pi}, a_i = a_i^j)$ poses exponential $(2^{l(\pi)})$ queries in the length of π (as opposed to a single query in absence of missing values). In general probabilistic imputation (without access to underlying data) is feasible only when $\Omega(T(D)$ is small and in practice may require use of optimization techniques.

2.2.5 Optimization Techniques

We now describe optimization techniques that can be used to reduce the query complexity of the Plan $P_{default}$ that has been introduced earlier to compute the necessary statistics required to build Naive Bayes and decision trees.

2.2.5.1 Optimization Techniques for Constructing Naive Bayes Classifier

For a Naive Bayes the plan $P_{default}$ poses all queries of the form $S(D)$, $S(D, C = c_k)$ and $S(D, C = c_k, a_i = a_i^j)$. Consider the case when D has no missing values. In such a case the results to the queries of the form $S(D, C = c_k)$ can be computed from the queries of form $S(D, C = c_k, a_i = a_i^j)$ as $\sum_{a_i^j \in V_{a_i}} S(D, C = c_k, a_i = a_i^j)$. Similarly, result of the query $S(D)$ can be computed as $\sum_{c_k \in V_c} \sum_{a_i^j \in V_{a_i}} S(D, C = c_k, a_i = a_i^j)$. However, in our setting, the dataset D cannot be inspected in a straightforward manner absence of missing values (as access to it is unavailable). Let $\delta a_i(D) = S(D) - \sum_{c_k \in V_c} \sum_{a_i^j \in V_{a_i}} S(D, C = c_k, a_i = a_i^j)$. Then if $\forall a_i \in V_{a_i}$, $\delta a_i = 0$, it implies D has no missing values (note this require the query for $S(D)$ to be explicitly submitted). Let the optimizations described above be named *opt1* and P_{opt1} denote an plan the improves the default plan $P_{default}$ using optimizations *opt1*. Hence, for a Naive Bayes and decision stump, $QC(P_{opt1}) = 1 + \sum_{a_i \in A} |V_{a_i}||V_C|$.

When the dataset D contains missing values, the optimization *opt1* cannot be applied. Consider case when D contains missing values, but there exists an attribute $a_1 \in V_{a_i}$ such that $\delta a_1(D) = 0$. In such case the result to $S(D, C = c_k)$ can be computed as $\sum_{a_1^j \in V_{a_1}} S(D, C =$

$c_k, a_1 = a_1^j)$. In addition, since D contains missing values, the plan $P_{default}$ to handle missing values (for either *mode imputation* or *probabilistic imputation*) needs to pose additional queries of the form $S(D, C = c_k, a_i =?)$. However, $S(D, C = c_k, a_i =?)$ can be computed as $S(D, C = c_k) - \sum a_i \in A \sum_{a_i^j \in V_{a_i}} S(D, C = c_k, a_i = a_i^j)$. In the case when there exists no attribute $a_1 \in A$ such that $\delta a_1(D) = 0$, to optimize the number of queries submitted to D, pose queries of the form $S(D, C = c_k)$ ($|V_C|$ in number) and then use them to calculate results to the queries of the form $S(D, C = c_k, a_i =?)$ ($|V_C| \times |A|$ in number). Let P_{opt1+} denote a plan that improves on the plan $P_{default}$ using the optimizations described above. Hence, for a Naive Bayes and decision stump, $QC(P_{opt1+})$ is $1 + \sum_{a_i \in A} |V_{a_i}||V_C|$ when $\exists a_1 \in A | \delta a_1(D) = 0$ and $QC(P_{opt1+})$ is $1 + |V_C| + \sum_{a_i \in A} |V_{a_i}||V_C|$ when $\forall a_1 \in A | \delta a_1(D)! = 0$.

2.2.6 Optimization Techniques for Constructing Decision Tree Classifier

For a decision tree, in absence of missing values in D, the results to the queries of the form $S(D^\pi, C = c_k)$, $S(D^\pi)$ and $S(D^\pi, a_i = a_i^j)$ can be computed from the results to the queries of the form $S(D^\pi, C = c_k, a_i = a_i^j)$ (on lines similar to described for Naive Bayes). In addition the queries posed to extend a node (corresponding to path π) can be computed from the results of queries needed to extend the siblings of π and of the parent (this optimization can be seen as getting the results to the queries needed to be posed to the last sibling node for free). Let the optimizations described by named *opt2* and P_{opt2} be a plan that improves $P_{default}$ by using optimizations *opt2*. For a decision tree $T(D)$, let $Z(D)$ be the set of nodes that is formed for $\Omega(T(D))$ by removing one child node for each no leaf node. Then $QC(P_{opt2}) = 1 + \sum_{\pi \in Z(T(D))} \sum_{V_{a_i} \in \mathcal{V}_\pi - \{V_C\}} |V_{a_i}| \times |V_C|$. Note to handle missing values by mode imputation, the query complexity of P_{opt2} will also be same (recall handling missing values by mode imputation has no additional penalty in terms of query complexity).

For handling missing values by probabilistic imputation, the plan $P_{default}$ to calculate the result to a query of the form $S(D^\pi, C = c_k, a_i = a_i^j)$ poses $2^{l(\pi)}$ primitive queries to D. Recall this is due to absence of access to underlying data D and hence inability to pose a query to D^π in a straightforward manner. Let $X_\pi = \{a_i | a_i \in A(\pi)\, and\, \delta a_i = 0\}$ be the set of attributes in π

that do not have any missing values in D. Hence query of form $S(D, C = c_k, a_i = a_i^j)clause(\pi_r)$ where π_r contains an fragment where an attribute in X_π takes value ? is zero and need not be posed. This optimization reduces the number of queries to calculate $S(D\pi, C = c_k, a_i = a_i^j)$ to $2^{l(\pi)} - 2^{|X_\pi|}$. This optimizations may make feasible *probabilistic imputation* under certain conditions (e.g. $\forall\pi, l(\pi)$ is close to $|X_\pi|$). Note optimization *opt*2 can be used once the values for $S(D^\pi, c_k, a_i = a_i^j)$ have been updated according to the technique described earlier to handle missing values.

2.3 Implementation and Results

The prototype implementation was written in Java and uses Java Database Connectivity (JDBC) API to establish communication with the relational database (e.g. MySQL) that contains the data set over which the predictive model is to be built. A class diagram of major classes in the implementation is shown in Figure 2.1. The implementation can be extended to add new classifiers. Any extension to the implementation that wants to add a new classifier needs to implement the interface *Classifier*. Any new type of data source (besides relational database) can be added by implementing the interface *SSDataSource*. The system also includes utilities to support learning from data contained in an Attribute-Relation File Format (ARFF)(an ARFF file is an ASCII text file that describes a list of instances sharing a set of attributes (see Witten and Frank (2005))). Essentially, this involves reading the instances in the ARFF file one at a time and inserting them in a database after which then learning proceeds as before (i.e. using SQL count queries). This approach allowed us to compare our system with WEKA [Witten and Frank (2005)] in a transparent way and in our experiments we were able to handle, in every case, datasets on which WEKA ran out of memory (the experiments were run on an Intel T2050 CPU T2050 with a 1.60 GHz processor and 1.24GB of RAM). This is an expected result since theoretically a sufficient statistics based approach to learning should be limited only by the size of the data repository (assuming there is enough RAM to hold the sufficient statistics).

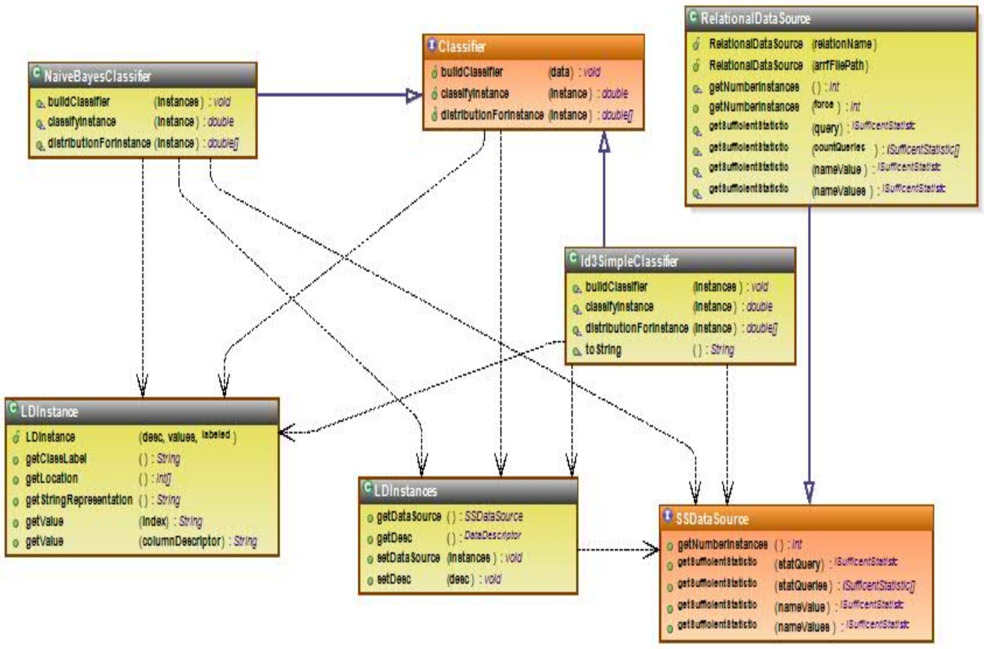
Classifier
buildClassifier (data) : void
classifyInstance (Instance) : double
distributionForInstance (Instance) : double[]
RelationalDataSource
RelationalDataSource (relationName)
RelationalDataSource (arffFilePath)
getNumberInstances () : int
getNumberInstances (force) : int
getSufficientStatistic (query) : ISufficentStatistic
getSufficientStatistic (countQueries) : ISufficentStatistic[]
getSufficientStatistic (nameValue) : ISufficentStatistic
getSufficientStatistic (nameValues) : ISufficentStatistic
NaiveBayesClassifier
buildClassifier (Instances) : void
classifyInstance (Instance) : double
distributionForInstance (Instance) : double[]
Id3SimpleClassifier
buildClassifier (Instances) : void
classifyInstance (Instance) : double
distributionForInstance (Instance) : double[]
toString () : String
LDInstance
LDInstance (desc, values, labeled)
getClassLabel () : String
getLocation () : int[]
getStringRepresentation () : String
getValue (index) : String
getValue (columnDescriptor) : String
LDInstances
getDataSource () : SSDataSource
getDesc () : DataDescriptor
setDataSource (Instances) : void
setDesc (desc) : void
SSDataSource
getNumberInstances () : int
getSufficientStatistic (statQuery) : ISufficentStatistic
getSufficientStatistic (statQueries) : ISufficentStatistic[]
getSufficientStatistic (nameValue) : ISufficentStatistic
getSufficientStatistic (nameValues) : ISufficentStatistic

Figure 2.1 Class diagram of core classes in ILF implementation.

2.4 Summary and Related Work

In this chapter we have presented a system that learns from data using only sufficient statistics. We described techniques for handling missing values and filters without the need to access underlying data or execution of user defined code on the data repositories. We presented an implementation of Naive Bayes and decision trees and precisely described the type of queries required to build these classifiers. We also studied the effect of missing values in the dataset on the number of queries required to build the Naive Bayes and decision tree classifier.

The related work in this area has primarily focused on scaling up to large data sets either by parallelizing specific algorithms [Jin and Agrawal (2003), Tveit and Engum (2003), Zaki (1999)], using distributed learning [Provost (2000), Provost and Kolluri (1999)] or providing support for disk resident data as in SPRINT [Shafer et al. (1996),Hsu et al. (2008)] and CLOUDS [Alsabti et al. (1998)]. Sufficient statistics based approaches to learn decision trees from large data sets have been suggested in [Moore and Lee (1998), Bar-Or et al. (2005)].

However, open source implementations of popular learning algorithms that scale to large data sets having been lacking. WEKA [Witten and Frank (2005)] an open source implementation of popular machine learning algorithms assumes in memory access to dataset and cannot scale to large data sets. WekaDB [Zou et al. (2006)] that described adding RDBMS support to WEKA does resemble our approach but our model differs in that it learns using only sufficient statistics and hence is applicable even in scenarios which preclude access to underlying data or execution of user supplied code at the individual data sources. Further we deal with data pre-processing (filters, handling missing values) at the statistics level rather than at instance level as is the case in WekaDB. There are several interesting directions for further development of the Indus Learning Framework. At the implementation level we assumed that all the attributes have a multi nominal distribution and need to extend it to case of attributes with continuous probability distributions (i.e. handle real valued attributes). One approach to handling real valued attributes may be to convert the continuous values into discrete values (see approach in Fayyad and Irani (1992)] but it will require the data source to provide additional statistics which can be used as cut-offs to convert the real values into binary bins. Further enhancements to the system include incorporating a data integration component that will allow learning from heterogeneous distributed data sources.

CHAPTER 3. Design and Implementation of a Query Planner for Data Integration

Emerging data-intensive applications e.g., in health informatics, security informatics, social informatics, etc. require integrated access to multiple distributed, autonomous, and often semantically disparate, data sources. Addressing this data integration challenge calls for techniques for bridging the semantic gap between the user and the data sources and for decomposing a user query into queries that can be processed by the individual data sources and for combining the results to produce the answer to the user query. This chapter describes the design and implementation of a system for data integration that solves these two problems.

3.1 Introduction

Recent advances in high throughput data acquisition technologies in many areas have led to a proliferation of a multitude of physically distributed, autonomous, and often semantically disparate data sources. Effective use of such data in data-driven knowledge acquisition and decision support applications e.g., in health informatics, security informatics, social informatics, etc. presents a *data integration* challenge. Addressing this data integration challenge requires techniques for bridging the semantic gap between the user and the data sources with respect to both the data schema and the data content (see Doan and Halevy (2005) for a survey). In a distributed setting, this also requires techniques for coping with *horizontal* and/or *vertical* data fragmentation. In the case of horizontal fragmentation, each data source (e.g., economic data for different states) contains a subset of data tuples that make up the data source of interest (e.g., economic data for the nation). In the case of vertical fragmentation, the different data sources contain subtuples of data tuples make up the data source of interest. Solving the data integration problem in such a setting presents us with a *query planning* problem, that is, the problem of decomposing a user query q into queries that can be processed (in the order specified by a *query plan*) by the individual data sources $D_1, D_2 \cdots D_p$ and for combining the results to produce the answer to the user query q. In general, there can be multiple query plans for answering a query query q from a collection of data sources $D_1, D_2 \cdots D_p$, with some plans being more optimal than others (e.g., the *cost* of execution). This chapter describes the design and implementation of a query planner for data integration that solves these two problems. The resulting query planner has been integrated into INDUS , an open source suite of algorithms for learning predictive models from autonomous, distributed, semantically disparate data sources in settings where it is not practical to access the underlying data in a centralized data warehouse.

We introduce the data integration problem through an example. Consider two universities U_1 and U_2 that collect information about their student employees (such as id, name, salary, the number of years in the university and academic status). Suppose at U_1, the registrar's office provides access to the collected information through a data source D_1. Suppose at U_2, similar information is gathered by the admissions department (which records the status of a student) in data source D_2. and a payroll department (records information such as salary, number of years in the university) in data source D_3. Further assume that the two universities use different underlying ontologies to describe student employees (see Figure 3.1). The schema associated with the data sources are given below :

1. Schema for D_1 is: *D1_Table(id(int), student-status(status-AVH), compensation(float), alias(varchar), serviceLength(int))*

2. Schema for D_2 is: *D2_Table(SSN(int), student-type(student-type-AVH))*

3. Schema for D_3 is: *D3_Table(social(int), salary(float), nickName(varchar), serviceYears(int))*

Let the ontologies for attribute *student-status* in D_1 and *student-type* in D_2 be as shown in Figure 3.1.

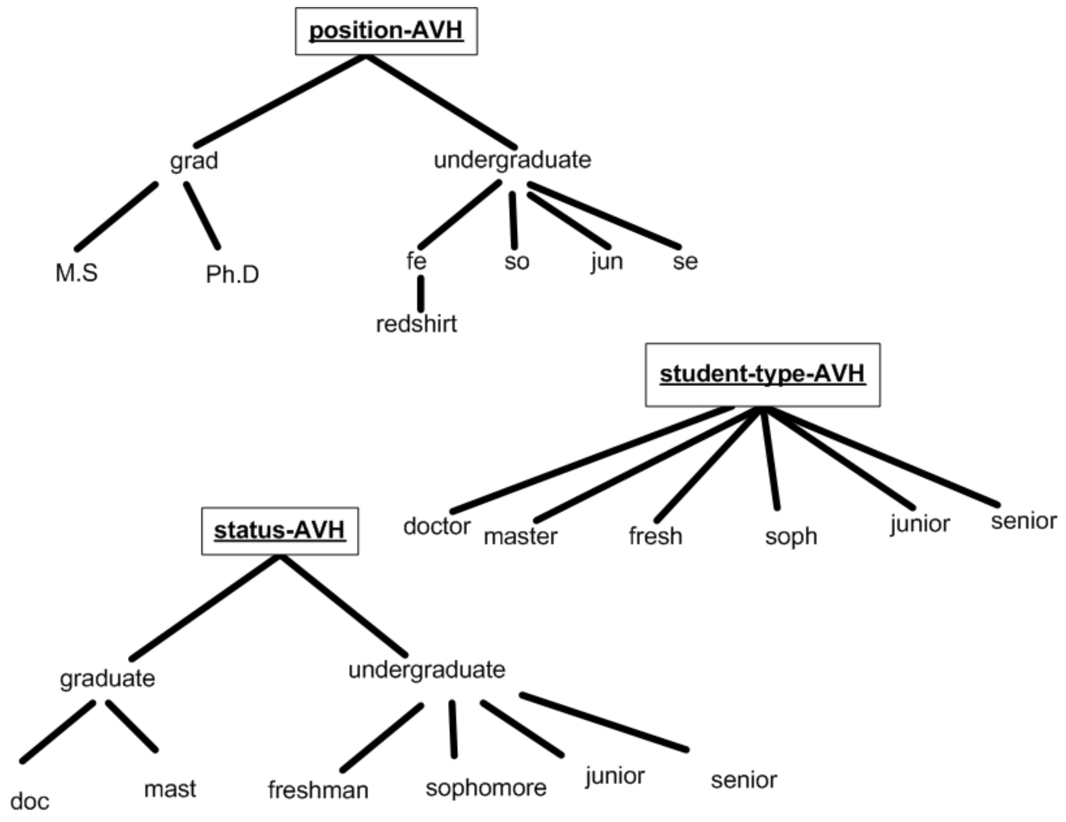

Figure 3.1 Ontologies associated with attributes position, student-status and student-type.

D_U	$\mapsto D_1$	$\mapsto D_2$	$\mapsto D_3$
key	id	SSN	social
position	status	type	-
benefits	compensation	-	salary
firstName	alias	-	nickName
timeHere	serviceLength	-	serviceYears

Table 3.1 Schema mappings for introduced example.

Consider a user, e.g. a statistician, interested in constructing predictive model based on the data from these two universities. Suppose the statistician needs integrated access to the data in D_1, D_2 and D_3 through a single (virtual) data source D_U with schema: *EMPLOYEETABLE*(*key*(*int*), *position*(*position-AVH*), *benefits*(*float*), *firstName*(*varchar*), *timeHere(int)*) with the ontology for attribute *position* as in Figure 3.1. The schema associated with D_1, D_2 and D_3 differ from each other and from the schema associated with D_U. The data sources describe student employees using different ontologies (e.g. *student-type, student-status* versus *position*). Furthermore, from the statistician's perspective, the dataset of interest is fragmented across the three data sources D_1, D_2 and D_3. Hence, when the statistician poses a query q against D_U, it needs to be translated to queries that can be answered by the individual data sources and the results appropriately combined to produce the answer to q. The query translation requires mappings from the schema and the the ontology associated with D_U to the schema and ontologies associated with D_1, D_2 and D_3. Suppose the schema mappings and ontology mappings be as specified in Table 3.1 and Table 3.2 respectively. Note that in general the mappings between ontologies may be incomplete. We now proceed to show how to solve the said data integration problem.

3.2 Solving the Data Integration Problem

3.2.1 Problem Description

We associate with each data source, a data source description (i.e., the schema and ontology of the data source) yielding an *ontology extended* data sources (OEDS). Formally An

status-AVH $\mapsto$ *position-AVH*	*student-type-AVH* $\mapsto$ *position-AVH*

junior = jun
graduate > M.S
 	
freshman = fe
	junior > redshirt
	fresh < undergraduate

Table 3.2 Mappings between ontologies associated with attribute *status*, attribute *position* and attribute *type*.

OEDS is a tuple $\mathcal{D} = \{D, S, O\}$, where D is the actual data set in the data source, S the data source schema and O the data source ontology Caragea et al. (2005). The formal semantics of OEDS are based on ontology-extended relational algebra Bonatti et al. (2003). Let $\mathcal{D}_1 = \{D_1, S_1, O_1\}, \mathcal{D}_2 = \{D_2, S_2, O_2\} \ldots \mathcal{D}_p = \{D_p, S_p, O_p\}$ be a set of p ontology extended data sources. Let $\mathcal{D}_U = \{D_U, S_U, O_U\}$ be a (virtual) integrated data source from the user's perspective. A user's perspective is specified by $P_U = \{D_U, M_U, \Psi_U\}$ where M_U is the set of mappings from the user schema S_U to the data source schema $S_1 \ldots S_p$ and Ψ_U a set of semantic correspondences from user ontology O_U to the data source ontologies $O_1, O_2 \ldots O_p$. For simplicity, we consider case when the schema mappings are one to one (i.e. every attribute in S_U has a corresponding attribute in schema S_i), the ontologies are attribute value hierarchies and the semantic correspondences take the form of $x < y$ (x is semantically subsumed by y i.e. a subclass relationship), $x > y$ (x semantically subsumes y i.e. a superclass relationship), $x = y$ (x is semantically equivalent to y i.e. equivalent class relationship) and the individual data sources are either horizontal or vertical fragments of the the data from the user's perspective. Let Q be the set of all possible queries that a user can pose against $\mathcal{D}_U$. We use an SQL like syntax (called SQL_{indus}) to describe the queries in Q. In SQL_{indus} a query $q \in Q$ is expressed as $\langle s, f, w \rangle$ where s is the set of attributes in the *Select* clause, f is the set corresponding to the tables in the *From* clause and w is an expression representing the *Where* clause. The clause w is expressed using the grammar $w = \epsilon | w_{atomic} | w \ AND \ \ w | w \ OR \ w$ and $w_{atomic} = column.name \ op_1 \ column.value$ where $op_1 \in \{> \mid < \mid = \mid != \}$. When the *column.name* has an ontology (attribute value hierarchy) associated with it, we overload the operators $>, <$ and $=$ to imply superclass, subclass and equivalent class respectively. The query $\langle s, f, \epsilon \rangle$ represents the case in which the *where* clause is absent. For aggregate queries we restrict our discussion to count queries as certain classifiers (e.g. Naive Bayes) over a dataset stored in a Relational Database Management Systems (RDBMS) can be learned using only SQL count queries (see Koul et al. (2008)). A data integration problem in this setting can be seen as the triple $\langle \mathcal{D}^*, P_U, Q \rangle$ where $\mathcal{D}^* = \{\mathcal{D}_1, \mathcal{D}_2 \ldots \mathcal{D}_p\}$, $P_U = \{D_U, M_U, \Psi_U\}$ is the user perspective and Q is the set of possible queries that can be posed by the user. We say that the data integration problem is *well-specified* if the data sources combine to form a user view of the data using the mappings specified (an arbitrary set of data sources and mappings may not form the user view of the data). In this thesis we restrict ourselves to *well-specified* data integration problems.

3.2.2 Overview of Solution

We proceed to describe a solution to the data integration problem using a data structure called *DTree*. A *DTree* is a binary tree in which the leaf nodes correspond to the actual data sources. Each internal node is a virtual data source that combines information from its two children. The structure of the *DTree* specifies the constraints on the order in which the data from the individual data sources are combined with the root node denoting a virtual data source that corresponds to D_U. A query against D_U is submitted to the root of the *DTree* and recursively divided into sub-queries such that the leaf nodes receive the queries to be executed against the respective individual data sources. The results from the leaf nodes are recursively combined up the tree with the root node receiving the answer to the query. The leaf nodes in *DTree* cope with the problem of bridging the semantic gap between the user and the data sources (see below) whereas the internal nodes cope with data fragmentation (by combing the results from their respective children). A *DTree* for the introduced example is shown in Figure 3.2.

3.2.3 *DTree*

Let $\Phi^{-1}(S_i)$ denote the set of attributes in S_U that are mapped to corresponding attributes in S_i (recall the schema mappings between S_U and S_i are one to one).

Definition *DTree:* A *DTree* $\tau(\pi)$ that solves a well-specified data integration problem $\pi = \langle \mathcal{D}*, P_U, Q\rangle$, is a **binary tree** with the following properties:

- There are $|\mathcal{D}^*|$ leaf nodes in the $\tau(\pi)$ and $\forall \mathcal{D}_i \in \mathcal{D}^*$ there is a leaf node labeled $\mathcal{D}_i$ and has an associated set $\Phi^{-1}(S_i)$.

- A non leaf node with children labeled $\mathcal{D}_i$ and $\mathcal{D}_j$ is labeled $\mathcal{D}_i\_\mathcal{D}_j$ and the associate set is $\Phi^{-1}(S_i) \cup \Phi^{-1}(S_j)$.

- For each non leaf node $\mathcal{D}_i\_\mathcal{D}_j$ either $\Phi^{-1}(S_i) = \Phi^{-1}(S_j)$ (i.e. horizontal fragmentation) or $\exists a \in \Phi^{-1}(S_i) \cap \Phi^{-1}(S_j)$ (i.e. vertical fragmentation) and a is the attribute over which $\mathcal{D}_i$ and $\mathcal{D}_j$ can be combined (join identifier).

- The associated set with the root node is S_U.

The procedure in *Algorithm 1* outlines the steps to construct a *DTree* $\tau(\pi)$ for a *well-specified* integration problem π.

Algorithm 1: *DTree* Construction.

```
Input:  A well-specified data integration problem $\pi = \langle \mathcal{D}^*, P_U, Q\rangle$
Output: A DTree $\tau(\pi)$ that solves $\pi$
$N \mapsto$ emptyset
$\forall \mathcal{D}_i = \langle D_i, S_i, O_i\rangle \in \mathcal{D}^*$ construct a node labeled $\mathcal{D}_i$ and associate with it the set
$\Phi^{-1}(S_i)$ and add it to $N$ .
repeat
    while  $\exists \mathcal{D}_i, \mathcal{D}_j \in N$ such that $\Phi^{-1}(S_i) = \Phi^{-1}(S_j)$ do
        begin
            Replace nodes $\mathcal{D}_i$ and $\mathcal{D}_j$ in $N$ by a node named $\mathcal{D}_i\_\mathcal{D}_j$ and associated it
            with the set $\Phi^{-1}(S_i)$
    if  $\exists \mathcal{D}_i, \mathcal{D}_j \in N$ and no $\mathcal{D}_k \in N$ such that $\Phi^{-1}(S_i) \cup \Phi^{-1}(S_k) \subseteq \Phi^{-1}(S_j)$ OR
    $\Phi^{-1}(S_j) \cup \Phi^{-1}(S_k) \subseteq \Phi^{-1}(S_i)$ then
        Replace nodes $\mathcal{D}_i, \mathcal{D}_j \in N$ by a node named $\mathcal{D}_i\_\mathcal{D}_j$ and associate it with the set
        $\Phi^{-1}(S_i) \cup \Phi^{-1}(S_j)$
until  until no change in $|N|$
```

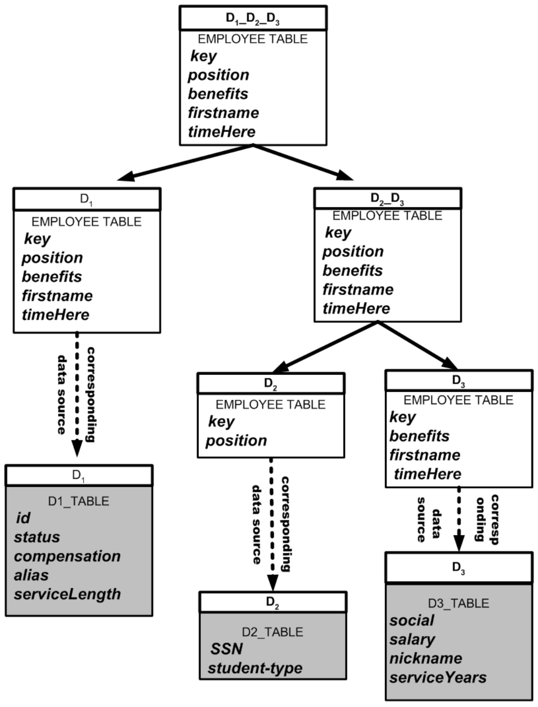

Figure 3.2 DTree for the motivating example.

It is easy to see that a *DTree* exists for a well specified data integration problem; and that in general, given a well specified data integration problem, there can be more than one *DTree* that solves it. Consider for example a dataset that has three horizontal fragments D_1, D_2 and D_3; There are two possible *DTree*s in this case: one that combines D_1 with D_2 first followed by D_3 and one that combines D_2 with D_3 first followed by D_1. Our algorithm outputs one of the possible *DTree*s that solves the given data integration problem. However, the algorithm can be modified to output an optimal *DTree* (based on some user-specified criteria).

3.2.4 Query Planner

A query posed by the user against D_U is submitted to the root node of the *DTree* $\tau(\pi)$ that solves the data integration problem $\langle \mathcal{D}^*, P_U, Q \rangle$. A query planner is invoked at each non leaf node of $\tau(\pi)$ to compute the set of plans that can be used to answer the query submitted to the node. Suppose a query q is submitted to a node n in the *DTree* $\tau(\pi)$. The task of the the query planner is is to output a *set* of plans P such that each $p \in P$ is of the form $\{q_n^l, q_n^r, \oplus\}$ where q_n^l and q_n^r are the queries submitted to the left and right child of node n and $\oplus$ is a binary operator applied to the results of q_n^l and q_n^r to obtain the results to q.

Intuitively, the operator $\oplus$ specifies how to aggregate the results obtained from the children of the current node and we refer to $\oplus$ as as *aggregation strategy.* For simplicity, we initially assume that there is no data overlap among the p data sources (we later show how to remove this constraint). Denoting by r_l and r_r the results obtained from the left and the right child of a node, we specify the following aggregation strategies:

$\sqcup$: $\sqcup(r_l, r_r)$ denotes a multiset of all the rows/tuples in r_l and r_r. We overload the operator $\sqcup$ to denote the addition of counts in the case of count queries.

$\bowtie$:$\bowtie_{id}$ (r_l, r_r) denotes an inner join of r_l and r_r on id.

$\uparrow$: denotes that there is no need for aggregation which is the case when a query is submitted to only one child.

$\langle \phi, q_{local} \rangle$ where $\phi \in \{\sqcup, \bowtie_{id}, \uparrow\}$: obtain $r_{temp} = \phi(r_l, r_r)$ and then obtain the final results by running the query q_{local} on r_{temp}.

$\langle \phi, q_{remote} \rangle$ where $\phi \in \{\sqcup, \bowtie_{id}, \uparrow\}$: obtain $r_{temp} = \phi(r_l, r_r)$. Use r_{temp} to construct query q_{remote} from a template and generate a new plan for this q_{remote}. This corresponds to a *two step plan* in which the results in the first step are used to compose the query q_{remote} for which a planner is again invoked.We now introduce some notation used in describing the Planner Algorithm. Given $\langle q, n, \tau(\pi) \rangle$ where $q = \langle s, f, w \rangle$, we define the following functions:

$sig(x)$ returns the set of attributes that appear in x where $x \in \{q, s, w\}$. For n , $sig(n)$ returns $\Phi^{-1}(n)$.

$\mathcal{F}_{join}(n)$ returns the join column for the children of n (applicable for vertical fragments only).

$\mathcal{F}^l_{child}(n)$ and $\mathcal{F}^r_{child}(n)$ returns the left and right child of the node n respectively.

$s^l_n = sig(s) \cap sig(\mathcal{F}^l_{child}(n))$ and $s^r_n = sig(s) \cap sig(\mathcal{F}^r_{child}(n))$ returns the select columns that are present in the left and right child of n respectively.

$q^+ = \langle s \cup \mathcal{F}_{join}(n), f, w \rangle$ adds a join column to the select clause of the query.

$\mathcal{T}_{allData}(q, n, l)$ $=\langle (sig(q^+) \cap sig(\mathcal{F}^l_{child}(n))), f, \epsilon \rangle$ retrieves the data corresponding to columns of the query q^+ that are present in the left child of n. Similarly we define $\mathcal{T}_{allData}(q, n, r) = \langle (sig(q^+) \cap sig(\mathcal{F}^r_{child}(n))), f, \epsilon \rangle$.

$Result(q)$ represents the result of the query q. $singlePath(q, n)$ returns true if all the columns in q are present in one child of n. WLOG (without loss of generality) we assume when the functions returns true, all the columns are present in $\mathcal{F}^l_{child}(n)$.
$horizontalFragmentation(n)$ returns true if $\mathcal{F}^l_{child}(n)$ and $\mathcal{F}^r_{child}(n)$ form the horizontal fragments of n. As an illustration for the motivating example it will return $true$ for node $D_1\_D_2\_D_3$ and $false$ for node D_2_D3.
For a node n we define a template for a where clause as $W^n_{temp} = \mathcal{F}_{join}(n)$ IN \$$values$\$. The function $Replace(W^n_{temp}, vals)$ replaces the place holder \$values\$ in the template W^n_{temp} by a comma separated list of values in $vals$. This template is used in a *two step plan*, where the results of the first step are used in the template to construct the query for the second step.

Once a query q is submitted to a node n in the *DTree*, the plan(s) generated to answer the query depends on how the attributes in s and w clause are distributed (based on how the data is fragmented) among the children of node n. We specify the the different data fragmentation scenarios using the function $\mathcal{M}(q, n) \mapsto \langle C_0, C_1, C_2, C_3\rangle$ where C_0 is set to 1 when the attributes of the select clause are distributed among the two children of the node. That is,

$$C_0 = \begin{cases} 1 & sig(s) \supset sig(\mathcal{F}^l_{child}(n))\ and \\ & sig(s) \supset sig(\mathcal{F}^r_{child}(n)) \\ 0 & \text{otherwise} \end{cases}$$

C_1 is set to 1 when the attributes in the where clause are distributed among the two children of the node.

$$C_1 = \begin{cases} 1 & sig(w) \supset sig(\mathcal{F}^l_{child}(n))\ and \\ & sig(w) \supset sig(\mathcal{F}^r_{child}(n)) \\ 0 & \text{otherwise} \end{cases}$$

In this case it has to be $w = w_l\, op\, w_r$ where $op \in \{AND, OR\}$. C_2 is set to 1 when $C_1 = 1$ and $w = w_l\, op\, w_r$ and the attributes in w_l and w_r occur individually in the two children. Assuming WLOG that the signature of the left child includes the signature of w_l whereas the signature of the right child includes the signature ofw_r, we have:

$$C_2 = \begin{cases} 1 & \mathit{when}\, C_1 = 1\, \mathit{and}\, w = w_l\, \mathit{op}\, w_r\, \mathit{and} \\ & \mathit{sig}(w_l) \subseteq \mathit{sig}(\mathcal{F}^l_{\mathit{child}}(n))\, \mathit{and} \\ & \mathit{sig}(w_r) \subseteq \mathit{sig}(\mathcal{F}^r_{\mathit{child}}(n)) \\ 0 & \text{otherwise} \end{cases}$$

C_3 is set to 1 when C_1 is 1, $C_2 = 0$ and $w = w_l\, op w_{rl}$ and the attributes in w_l occurs completely in one child and attributes in w_{rl} are distributed over the two children. Assuming WLOG that the signature of the left child includes the signature of w_l we have:

$$C_3 = \begin{cases} 1 & \mathit{when}\, C_1 = 1, C_2 = 0\, \mathit{and}\;\; w = w_l\, \mathit{op}\, w_{rl}\, \mathit{and} \\ & \mathit{sig}(w_l) \subseteq \mathit{sig}(\mathcal{F}^l_{\mathit{child}}(n))\, \mathit{and} \\ & \mathit{sig}(w_{rl}) \supset \mathit{sig}(\mathcal{F}^l_{\mathit{child}}(n))\, \mathit{and} \\ & \mathit{sig}(w_{rl}) \supset \mathit{sig}(\mathcal{F}^r_{\mathit{child}}(n))\, \mathit{and} \\ 0 & \text{otherwise} \end{cases}$$

The value $\langle C_1, C_2, C_3 \rangle$ describes how the attributes in the *where* clause w are fragmented among the two children of the current node. The value $\langle 0, 0, 0 \rangle$ corresponds to *no fragmentation*, $\langle 1, 1, 0 \rangle$ corresponds to *clean fragmentation*, $\langle 1, 0, 1 \rangle$corresponds to *partial fragmentation* and $\langle 1, 0, 0 \rangle$ corresponds to *full fragmentation.* It follows from the definitions above that no other values of $\langle C_1, C_2, C_3 \rangle$ are possible. Since our goal is to effectively *minimize* data fragmentation, among the multiple equivalent ways of expressing w, we choose one that corresponds to the least amount of fragmentation (*no fragmentation* is preferred over *clean fragmentation* which in turn is preferred over *partial fragmentation* which in turn is preferred over *full fragmentation*). This is achieved by the function $DeFrag(n, w)$ which returns an equivalent *where* clause for w that has the least fragmentation. Consider for example, a query submitted to the node $D_2\_D_3$ in the DTree in Figure 3.2 with the *where* clause being (*position='Ph.D' AND benefits > 300*) *OR* (*benefits > 300 AND timeHere > 2*). This form of the *where* clause corresponds to partial fragmentation. However, $DeFrag$ function produces an equivalent *where* clause, (*position='Ph.D' AND timeHere > 2*) *OR* (*benefits > 300*) which corresponds to clean fragmentation. Using the notation introduced above, the *Query Planner* is outlined in *Algorithm 2*.

```
Algorithm 2: Query Planner
  Input:  $q$ posed to a node $n$ in $DTree$ $\tau(\pi)$ such that $sig(q) \supset sig(n)$
  Output: Set of Plans to answer $q$
  if horizontalFragmentation(n) then
      Add Plan $\mapsto q_l^n = q_r^n = q$; $\oplus = \sqcup$;
      Add Plan $\mapsto$ DefaultPlan($q, n$)
  else
      $\langle C_0, C_1, C_2, C_3 \rangle = \mathcal{M}(q, n)$ ; $w = DeFrag(n, w)$
      if $C_0 == 0$ then
          switch $\langle C_1, C_2, C_3 \rangle$ do
              case $\langle 0, 0, 0 \rangle$
                  Plans $\mapsto$ NoFragmentation($q, n$)
              case $\langle 1, 1, 0 \rangle$
                  Plans $\mapsto$ CleanFragmentation($q, n$)
              case $\langle 1, 0, 1 \rangle$
                  Plans $\mapsto$ PartialFragmentation($q, n$)
              case $\langle 1, 0, 0 \rangle$
                  Plans $\mapsto$ DefaultPlan($q, n$)
              otherwise
                  throw Exception("Not a Possible Case")
      else
          Plans $\mapsto$ SelectFragmented($q, n$)
```

Function DefaultPlan(q, n)

```
$q_l^n = \langle \mathcal{T}_{allData}(q, n, l) \rangle$; $q_r^n = \langle \mathcal{T}_{allData}(q, n, r) \rangle$
if $horizontalFragmentation(n)$ then
    $\oplus = \langle \sqcup, q_{local} \rangle$; where $q_{local} = q$;
else
    $\oplus = \langle \bowtie_{\mathcal{F}_{join(n)}}, q_{local} \rangle$ where $q_{local} = q$;
```

Function NoFragmentation(q, n)

```
if $singlePath(q,n)$ then
    Add Plan $\mapsto$ $q_l^n = q$; $q_r^n = null$; $\oplus = \uparrow$
else
    /*WLOG assume attributes in $s$ in right child and $w$ in left child    */
    Add Plan $\mapsto$
    $q_l^n = \langle \mathcal{F}_{join}(n), f, w \rangle$; $q_r^n = null$;
    $\oplus = \langle \uparrow, q_{remote} \rangle$ $q_{remote} = \langle s, f, Replace(W_{temp}^n, Result(q_l^n)) \rangle$;
    Add Plan $\mapsto$
    $q_l^n = \langle \mathcal{F}_{join}(n), f, w \rangle$; $q_r^n = \langle s \cup \mathcal{F}_{join}(n), f, \epsilon \rangle$; $\oplus = \{ \bowtie_{\mathcal{F}_{join(n)}} \}$;
```

Function CleanFragmentation(q, n)

```
$q = \langle s, f, w_l \; op \; w_r \rangle$
Add Plan $\mapsto$
$q_l^n = \langle \mathcal{F}_{join}(n), f, w_l \rangle$; $q_r^n = null$; $\oplus = \langle \uparrow, q_{remote} \rangle$
$q_{remote} = \langle s, f, w_r \; op \; Replace(W_{temp}^n, Result(q_l^n)) \rangle$
Add Plan $\mapsto$
Mirror of Plan Above (switch $l$ and $r$ in Plan Above )
Add Plan $\mapsto$
$q_l^n = \langle \mathcal{F}_{join}(n), f, w_l \rangle$; $q_r^n = \langle \mathcal{F}_{join}(n), f, w_r \rangle$;
**if** $op == AND$ **then** $\phi = \sqcap$; **else** $\phi = \sqcup$;
$r_{temp} = \phi(Result(q_l^n), Result(q_r^n))$;
$\oplus = \langle \phi, q_{remote} \rangle$;
$q_{remote} = \langle s, f, Replace(W_{temp}^n, r_{temp}) \rangle$
Add Plan $\mapsto$ /*applicable if $op$ is AND                                */

**if** $op == AND$ **then**
    $q_l^n = q^+$; $q_r^n = \langle \mathcal{F}_{join(n)}, f, w_r \rangle$; $\oplus = \{\bowtie_{\mathcal{F}_{join}(n)}\}$;
Add Plan $\mapsto$ DefaultPlan($q, n$);
```

```
Function PlansSelectFragmented($q, n$)
  $\langle C_0, C_1, C_2, C_3 \rangle = \mathcal{M}(q, n)$
  /*Function only called when $C_0 == 1$                                  */
  switch $\langle C_1, C_2, C_3 \rangle$ do
      case $\langle 0, 0, 0 \rangle$
          Add Plan $\mapsto$
          $q_l^n = \langle s_n^l, \cup\mathcal{F}_{join(n)}, f, w \rangle$; $q_r^n = \langle s_n^r, \cup\mathcal{F}_{join(n)}, f, \phi \rangle$; $\oplus = \langle \bowtie_{\mathcal{F}_{join(n)}}, q_{local} \rangle$;
          $q_{local} = q$;
          Add Plan $\mapsto$
          Assuming $w$ occurs completely in left child. $q_l^n = \langle \mathcal{F}_{join}(n), f, w \rangle$; $q_r^n = null$;
          $\oplus = \langle \uparrow, q_{remote} \rangle$; $q_{remote} = \langle s, f, Replace(W_{temp}^n, Result(q_l^n)) \rangle$;
          Add Plan $\mapsto$ DefaultPlan($q, n$)
      case $\langle 1, 1, 0 \rangle$
          Add Plan $\mapsto$
          $q = \langle s, f, w_l op w_r \rangle$
          if $op == AND$ then $\phi = \sqcap$;  else $\phi = \sqcup$;
          $q_l = \langle \mathcal{F}_{join}(n), f, w_l \rangle$; $q_r = \langle \mathcal{F}_{join}(n), f, w_r \rangle$; $\oplus = \langle \phi, q_{remote} \rangle$;
          $r_{temp} = \phi\left(Result(q_l^n, Result(q_r^n))\right)$; $q_{remote} = \langle s, f, Replace(W_{temp}^n, r_{temp}) \rangle$;
          Add Plan $\mapsto$ DefaultPlan($q, n$);
      case $\langle 1, 0, 1 \rangle$
          Add Plan  $\mapsto$ DefaultPlan($q, n$);
      case $\langle 1, 0, 0 \rangle$
          Add Plan $\mapsto$ DefaultPlan($q, n$);
      otherwise
          throw Exception("Not Applicable");
```

Function PartialFragmentation(q, n)

```
Add Plan ↦ DefaultPlan($q, n$);
Add Plan ↦
$q = \langle s, f, w_l \; op \; w_{rl} \rangle$.
if ($w_{rl} == w_{ll} \; op_2 w_{rr}$ such that $w_{rr}$ occurs completely in right child and $w_{ll}$ occurs
completely in left child, AND $s$ occurs completely in left child) then
    $w = (w_l \; op \; w_{ll}) op_2 (w_l \; op_1 \; w_{rr})$
    /*$op$ and $op_2$ are different as otherwise $DeFrag$ operation would have
      resulted in clean fragmentation                                          */
    if $op == AND$ then $\phi = \sqcap$; else $\phi = \sqcup$; $q_l^n = \langle \mathcal{F}_{join}(n), f, w_l \rangle$;
    $q_r^n = \langle \mathcal{F}_{join}(n), f, w_{rr} \rangle$; $\oplus = \langle \phi, q_{remote} \rangle$; $r_{temp} = \phi(Result(q_l^n), Result(q_r^n))$;
    $q_{remote} = \langle s, f, (w_l \; op w_{ll}) \; op_2 \; Replace(W_{temp}^n, r_{temp}) \rangle$;
```

The correctness of the plans generated by the query planner follows from the manner in which the results are combined at each node of the *DTree*. It can further be shown that the plans produced by the query planner ensure that each data source is queried at most twice. This follows from the observation that a data source is queried more than once when the aggregation strategy is $\langle \phi, q_{remote} \rangle$ (recall this aggregation strategy corresponds to a *two step plan* in which the results in the first step are used to compose the query q_{remote} for which a planner is again invoked). An straightforward analysis of the Planner algorithm shows that q_{remote} constructed in the first step, in each case, is such that any possible plan for it never has an aggregation strategy of the type $\langle \phi, q_{remote} \rangle$. Hence, a data source is queried at most twice to obtain a result to a query.

The preceding description of the Query Planner assumes that the data sources do not overlap. When this is not the case, we assume the existence of a primary key *key* in the schema S_U and that there is an one-to-one mapping between domain of *key* and the domain of the attribute that *key* is mapped to (via schema mappings) in each of the data sources; and user query $q = \langle s, f, w \rangle$ is changed to $\langle s \cup key, f, w \rangle$ before it is submitted to the root node of the DTree. The primary key allows the algorithm to detecting and dealing with the data tuples that are duplicated among the data sources.

Note that for a given user query q, the query planner is executed (and produces a *set* of query plans) at each node in the *DTree.* Answering the query q requires choosing one such plan at each of the non leaf nodes of the *DTree.* This choice can be made so as to optimize some desired criterion (e.g., estimated cost of answering the query q for each possible choice of plans).

3.2.5 Query Binding

Once a query plan is chosen for each node of the *DTree*, answering the user query reduces to recursively combining the answers to the sub queries that are passed to the leaf nodes of the *DTree* . To bridge the semantic gap between D_U and the individual data sources, sub query received by each leaf node needs to be expressed in a form that can be executed against the corresponding data source. We call this process *Query Binding.* It consists of three steps:. *Translation*, *Renaming* and *Rewriting.* Let $\Phi^{\leftarrow}(S_i)$ be a renaming of the schema S_i by using the the one to one schema mapping between S_U and S_i. The process of *Translation* converts a query q in schema S_U and ontology O_U into a query against schema $\Phi^{\leftarrow}(S_i)$ and ontology O_i. Interested readers are referred to Bao et al. (2007) for details of the translation process. The process of *Renaming* converts the translated query q_1 (now in schema $\Phi^{-1}(S_1)$) to a query q_2 that is against the schema S_i of the corresponding data source D_i . This process is straightforward since $\Phi^{-1}(S_1)$ is a sub-schema of S_U (by construction) and the mapping (M_U) between S_U and S_i is one to one. After Renaming the query q_2 is still in SQL_{indus} syntax and may include ontological relations (e.g. subclass and superclass) in the *where* clause of the query. The process of *Rewriting* converts the query q_2 in SQL_{indus} syntax into a query q_3 in the language understood by the data source while preserving the query semantics. For RDBMS data source the only non trivial operation in this process is to convert the subclass, superclass and equivalentclass relations (within the corresponding data source ontology) that appear in the *where* clause w (if any) of the query in SQL_{indus} syntax into appropriate expressions in SQL. We do this as follows: $\forall w_{atomic} = column.name\ op_1\ value$ where the attribute *column.name*

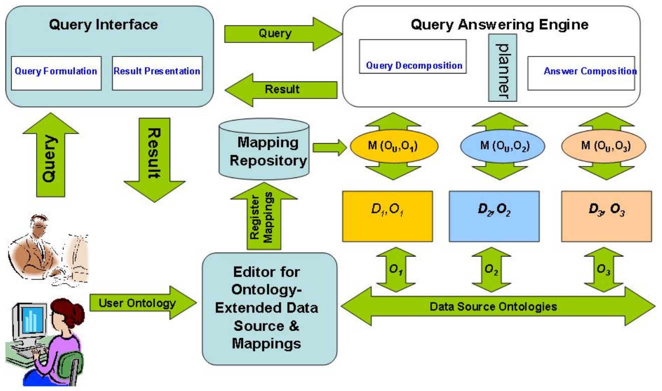

Figure 3.3 Indus with query planner.

has an AVH associated with it, replace w_{atomic} by a SQL fragment of the form *column.name IN valueSet* where $valueSet = \{x|x \in O_i$ and $x\ op_1\ value\ is\ true\}$. Thus, $status > freshman$ in SQL_{indus} posed against D_1 in the example shown in Section 1 is rewritten as: *status IN* $\{$*' undergraduate'*$\}$.

3.3 Implementation and Results

The proposed approach to data integration has been implemented in Java as part of INDUS, a system that we developed in our lab to enable an application to query a collection of semantically disparate relational data sources from a user's perspective. The overall architecture of the system is shown in Figure 3.3. The data sources are assumed to be *ontology extended* wherein an ontology is associated with each attribute in the data source. A mapping repository holds the schema mappings between the user view and the data source view as well as inter ontology mappings. The query answering engine (which incorporates the planner) is responsible for generating plans for a user submitted query, select a plan for execution and return results to the user. The current implementation restricts ontologies to be attribute value hierarchies, schema mappings as one to one and *a priori* selects a plan for execution (among multiple available plans).

The output (executed plan) for the query *select firstname from EMPLOYEETABLE where* *position* $>$ *'redshirt';* in the example described in Section 1 is shown below.

node $D_1\_D_2\_D_3$ plan:

q (query to node): *select firstname from EMPLOYEETABLE where position* $>$ *'redshirt';*

q_n^l (sub-query for node D_1): *select firstname from EMPLOYEETABLE where position* $>$ *'red-shirt';*

q_r^l (sub-query for node $D_2\_D_3$): *select firstname from EMPLOYEETABLE where position* $>$ *'redshirt';*

$\oplus$ (Aggregation Strategy) : $\sqcup$

node $D_2\_D_3$ plan

q (query to node): *select firstname from EMPLOYEETABLE where position* $>$ *'redshirt';;*

q_n^l (sub-query for node D_2): *select key from EMPLOYEETABLE where position* $>$ *'redshirt';*

q_r^l (sub-query for node D_3): *select firstname, key from EMPLOYEETABLE;*

$\oplus$ (Aggregation Strategy) : $\bowtie_{key}$

After query binding for D_1: *select alias from D1_Table where student-status IN ('undergradu-ate', 'freshman');*

After query binding for D_2 : *select ssn from D2_Table where student-type IN ('junior');*

After query binding for D_3: *select nickname, social from D3_Table;*

Similarly the executed plan for the count query *select count(firstname) from EMPLOYEETABLE where position* $>$ *'redshirt';* is shown below

node $D_1\_D_2\_D_3$ plan:

q (query to node): *select count(firstname) from EMPLOYEETABLE where position* $>$ *'red-shirt';*

q_n^l (sub-query for node D_1): *select count(firstname) from EMPLOYEETABLE where (position* $>$ *'redshirt');*

q_r^l (sub-query for node $D_2\_D_3$): *select count(firstname) from EMPLOYEETABLE where (po-sition* $>$ *'redshirt');*

$\oplus$ (Aggregation Strategy) : $\sqcup$

node $D_2\_D_3$ plan:

q (query to node): *select count(firstname) from EMPLOYEETABLE where position > 'red-shirt';*

q^l_n (sub-query for node D_2): *select key from EMPLOYEETABLE where (position > 'redshirt');*

q^l_r (sub-query for node D_3): no query submitted

$\oplus$ (Aggregation Strategy) : $\uparrow$

After query binding for D_1: *select count(alias) from D1_Table where (student-status IN ('un-dergraduate', 'freshman'));*

After query binding for D_2: *select ssn from D2_Table where (student-type IN ('junior'));*

After query binding for D_3: no query submitted

Interested readers are referred to the implementation of the system, open sourced at Koul (2008a), that includes JUNIT test cases for some additional queries that can be posed against the example introduced in Section 1. The implementation when used with the Indus Learning Framework (see Koul et al. (2008), Koul (2008b)), an open source implementation of an approach to learn classifiers from a data source using SQL count queries , enables the Indus Learning Framework to learn classifiers from multiple semantically disparate data sources.

3.4 Summary and Related Work

Summary: Emerging data-intensive applications e.g., in health informatics, security informatics, social informatics, etc. require integrated access to multiple distributed, autonomous, and often semantically disparate, data sources. Addressing this data integration challenge calls for techniques for bridging the semantic gap between the user and the data sources and for decomposing a user query into queries that can be processed by the individual data sources and for combining the results to produce the answer to the user query. This chapter describes the design and implementation of a system for data integration that solves these two problems. The resulting query planner has been integrated into INDUS, an open source suite of algorithms for learning predictive models from autonomous, distributed, semantically disparate data sources in settings where it is not practical to access the underlying data in a centralized data warehouse.

Related Work: The problem of data integration (see Lenzerini (2002) for theoretical overview) has received significant attention in literature (see Doan and Halevy (2005) for a survey). Most of this work has focused on bridging semantic differences between schema and ontologies associated with the individual data sources (see Shvaiko and Euzenat (2005),Wache et al. (2001) for surveys of approaches that address schema heterogeneity). Aspects of data content heterogeneity are addressed in Wache and Stuckenschmidt (2001), Goh et al. (1999). The approach in SIRUP [Ziegler and Dittrich (2004)] resembles our approach since the ontologies are added on top of data. However, the ontologies in SIRUP, built with *IConcepts*, are data schema ontologies. Users are required to build semantic perspectives from a selected subset of the data source *IConcepts*. As such SIRUP primarily addresses schema heterogeneity and does not need explicit mappings to convert from user view to a data source view. The handling of data content heterogeneity in COIN [Goh et al. (1999)] is limited to unit conversions (e.g.,dollars into euros) and term substitutions. BUSTER [Visser et al. (2000)] handles both schema and data heterogeneity but assumes an existence of a global ontology with each data source ontology being a refinement of the global ontology. In contrast our solution focuses on integrating ontology extended data sources that are fragmented and semantically heterogeneous from a user point view. It does not assume existence of a global ontology and uses mappings to handle schema and data content heterogeneity.

CHAPTER 4. Scalable, Updatable Predictive Models for Sequence Data

The emergence of data rich domains has led to an exponential growth in the size and number of data repositories, offering exciting opportunities to learn from the data using machine learning algorithms. In particular, sequence data is being made available at a rapid rate. In many applications, the learning algorithm may not have direct access to the entire dataset because of a variety of reasons such as massive data size or bandwidth limitation. In such settings, there is a need for techniques that can learn predictive models (e.g., classifiers) from large datasets without direct access to the data. We describe an approach to learn from massive sequence datasets using statistical queries. Specifically we show how Markov Models can be constructed from sequence databases that answer only count queries. We analyze the *query complexity* (a measure of the number of queries needed) for constructing classifiers in such settings and outline some techniques to minimize the query complexity. We also outline how a Markov Model can be updated in response to addition or deletion of subsets of sequences from the underlying sequence database.

4.1 Introduction

Advances in high throughput sequencing and other data acquisition technologies have resulted in gigabytes of DNA, protein sequence data, and gene expression data being gathered at steadily increasing rates. These developments have resulted in unprecedented opportunities for learning from such data. Most machine learning techniques assume direct access to data. However, in many practical applications, the massive size of the data being made available coupled with memory and bandwidth constraints prohibit direct access to data. In addition, it is not difficult to envision settings in the near future, such as personalized medicine where privacy concerns may prohibit direct access to the data (e.g. DNA sequence of patients under treatment). Further, in settings where data is being made available at a rapid rate (e.g. sequence data), a local copy of the data may quickly become out of date. Hence, there is an urgent need for approaches to learning predictive models, from large datasets (that cannot fit in the memory available on the device where the learning algorithm is executed), that are scalable, able to cope with frequent data updates and do not require access to the underlying dataset.

Caragea et al. (2004b) have introduced a general strategy for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries or procedures that can be executed on the remote data sources. This involves separating a learning algorithm into two components: (i) a statistical query [1] generation component that poses a set of statistical queries to be answered by a data source and (ii) a hypothesis construction component that uses the resulting statistics to modify a partially constructed hypothesis (and may further invoke the statistical query component as needed). Inspired by this work we extend this strategy to the setting of building predictive models from large sequence datasets by interacting with the data source that holds the dataset only through means of certain count queries. This approach allows us to cope with the challenges of massive data size (since in general the statistics of the data are much smaller than the size of the data), no access to underlying dataset (because it interacts with data source only through statistical queries) and in certain cases, data source updates (additions, deletions of large subsets of data).

We focus our attention on a class of *Markov Property* based class of predictive models for sequences: Markov Models, Probabilistic Suffix Trees, Interpolated Markov Models that are among some of the most widely used in sequence classification (e.g. Bejerano and Yona (2001)), text analysis (e.g. McCallum et al. (2000)) and related applications. We describe the specific type of queries that the data source should answer in order to build the predictive model, and precisely calculate the number of queries that are posed to a data source to build

[1]A statistic is simply a function of a dataset; A statistical query returns a statistic (e.g., the number of instances in the dataset that have a specified value for a specified attribute.)

the predictor. The number of queries posed (called the *query complexity* in our model) is a measure of steps required to build the model and may be important in the cases where the data source associates a cost with answering a query or in the setting where bandwidth is at a premium. We describe certain optimization techniques that can be used to minimize the query complexity and in particular describe a lazy approach to classifying a test dataset that can be used to ameliorate the exponential query complexity associated with a Markov Model. The rest of the chapter is organized as follows. Section 4.2 covers the preliminaries. Section 4.3 describes a statistical query based approach to constructing Markov Model based sequence classifier and outlines some optimization techniques to minimize the query complexity. Section 4.4 describes an extension of this approach to Probabilistic Suffix Trees. Section 4.5 describes an approach to updating Markov model based predictors using statistical queries. Section 6.5 concludes with a brief summary and description of related work.

4.2 Preliminaries and Notation

Let Σ be the alphabet from which the sequences are constructed and C be the set of classes to which the sequences can be assigned. Given a sequence $s = \sigma_1\sigma_2 \ldots \sigma_n$ and a symbol $\sigma \in \Sigma$, let $s\sigma$ represent the sequence $\sigma_1\sigma_2 \ldots \sigma_n\sigma$, let σs represent the sequence $\sigma\sigma_1\sigma_2 \ldots \sigma_n$ and $suffix(s)$ represent the sequence $\sigma_1\sigma_2 \ldots \sigma_{n-1}$. We associate with each dataset D of sequences a descriptor $Desc^s(D) = \langle \Sigma, C \rangle$ where Σ is the alphabet from which the sequences in D are constructed and C is the classes (e.g. protein family classes) to which the sequences in D can be assigned. Let $P(s)$ be the probability of observing a sequence s and $P(\sigma|s)$ be the probability of observing the symbol σ right after the subsequence s. The empirical values for $P(s)$ and $P(\sigma|s)$ are represented by $\hat{P}(s)$ and $\hat{P}(\sigma|s)$ respectively.

Suppose the data source D supports a set of primitive queries Q_D expressed in a query language supported by the data source holding D (e.g. if D is a RDBMS such as Oracle the query language will be SQL). To build a predictive model, we assume that the system expresses statistical queries against D in its own statistical query language Λ. A *query planner* Π that transforms a query $q(s_D)$ expressed in Λ for a statistic s_D into a plan for answering s_D using some subset of the primitive statistical queries Q_D. We assume that the query planner Π has at its disposal, a set of operators O that can be used to combine the answers to queries in Q_D to obtain a statistic s_D. In the case where Q_D correspond to count queries, O may include $+, -$. A *query plan* for s_D, denoted by $plan(s_D)$, is simply an *expression tree* that successively combines the answers to the primitive queries to obtain the answer to query s_D (expressed in the query language that is understood by the query planner): Each leaf node corresponds to a *primitive query* in Q_D and each non-leaf nodes corresponds to an operator in O. We assume that the planner Π is guaranteed to produce a correct plan $plan(s_D)$ for every statistic s_D that is expressible in Λ.

The learning algorithm L (say PST), when executed against a dataset D, generates at each step i, a *set* of statistical queries $S_i(D) = \{s_D(i,1) \cdots s_D(i,n_i)\}$ where each query in S_i is expressed in Λ. Let $Plan(S_i(D)) = \{plan(s_D(i,1)) \cdots plan(S_D(i,n_i))\}$ be the set of plans generated by the query planner for the set of queries $S_i(D)$. We denote by $Q(plan(s_D(i,j)))$, the set of the primitive queries used in the plan $plan(s_D(i,j))$. Note that $Q(Plan(S_i(D)))$ denotes the subset of *primitive* queries against D that to answer the set of queries $S_i(D)$. Let $Q(Plan(S_i(D))) = \sum_{j=1}^{n_i} Q(plan(s_D(i,j)))$. Let $Q^L = \sum_i Q(Plan(S_i(D)))$. Clearly, $\forall j\, Q(plan(s_D(i,j))) \subseteq Q(Plan(S_i(D))) \subseteq Q^L \subseteq Q_D$. Consider a sequence of sets of statistical queries $S_1(D) \cdots S_i(D)$ generated by L when it is executed against a dataset D. Let ϕ_i be the corresponding sequence of sets of query plans $Plan(S_1(D)), Plan(S_2(D)) \cdots Plan(S_i(D))$ produced by the query planner. Let $\hat{Q}(\phi_i) = \cup_{l=1}^{i} \hat{Q}(Plan(S_l(D)))$ denotes the set of primitive queries retrieved as a result. Assuming that L generates a sequence of m query sets $S_1(D) \cdots S_m(D)$ prior to terminating with a learned hypothesis, we can define the *query complexity* of ϕ_m, denoted by $QC(\phi_m)$, as $\left|\hat{Q}(\phi_m)\right|$, that is the total number of primitive queries that are posed to the data source based on ϕ_m. The task of the query planner is to generate a sequence of sets of query plans ϕ_m so as to minimize the query complexity $QC(\phi_m)$ which can be important in settings where the data source imposes a cost for answering each primitive query .

4.3 MM(k-1): Markov Model of order k -1

Markov Models have been used successfully in literature to address sequence based tasks (see Borodovsky and McIninch (1993), Burge and Karlin (1997), Begleiter et al. (2004)). Often it is assumed that the learner that builds the Makov Model has access to the training dataset D. In many applications, the learning algorithm may not have direct access to the entire dataset because of massive size of data, access restrictions, or bandwidth requirements. To address this setting, we assume that local acess to D is unavailable. However, we assume that the learner has access to the descriptor of the data (i.e. $Desc^s(D) = \langle \Sigma, C \rangle$) and the data source holding D answers certain count queries over the dataset D. In particular, we assume the data source answers the following three types of queries: (1) the query to compute the count of sequences in D that have the subsequence s (including overlaps), denoted by $S(D, s)$; (2) the query to compute the count of the sequences in D that belong to the class c_k and have the subsequence s (including overlaps), denoted by $S(D, s, C = c_k)$ and (3) the query to compute the count of sequences in D that belong to the class c_k and have subsequences of length $|s|$ (including overlaps), denoted by $S(D, |s|, C = c_k)$.

In the MM(k-1), the estimate of the probability that a given sequence (unlabeled) $s = \sigma_1\sigma_2 \ldots \sigma_n$ belongs to the class c_j is given by

$$\hat{P}_{MM(k-1)}(s, c_j) = \prod_{i=k}^{n} \hat{P}(\sigma_i | \sigma_{i-1} \ldots \sigma_{i-k+1}, C = c_j) \tag{4.1}$$

In the sufficient statistics model, the required terms in equation (4.1) can be computed using the supported queries as

$$\hat{P}(\sigma_i | \sigma_{i-1} \ldots \sigma_{i-k+1}, C = c_j) = \frac{S(D, \sigma_{i-k+1} \ldots \sigma_{i-1}\sigma_i, C = c_j)}{S(D, |s|, C = c_j)} \tag{4.2}$$

The Naive Bayes for sequence classification is special case of the MM(k-1) with $k = 1$ and as such can be implemented in the sufficient statistics model in a straightforward way. A straightforward approach to classify any given sequences using $MM(k-1)$ would be to pre-compute the results for all possible queries that may be needed to classify the set of sequences in Σ^*. It is clear from equations (4.1) and (4.2) that this involves all queries of the form

$S(D, s, C = c_j)$ where $|s| = k$ and $S(D, |s|, C = c_j)$ that can be posed over $Desc^s(D)$. Since the total number of unique subsequences of length (k) is $|\Sigma|^k$, the Query Complexity of this approach is $|C|(|\Sigma|^k + 1)$. As, the number of queries posed in this approach is exponential in k, it is often not feasible for large k. Hence, it is of interest to explore optimization techniques that minimize the number of queries posed to the data source D.

4.3.1 Optimization Techniques for MM(k-1)

In what follows, we give some examples of optimizations that can help reduce the query complexity of sequence classification. Let $\Sigma^k \subset \Sigma^*$ be the set of all possible sequences of length k over the alphabet Σ. It follows that $S(D, |s|, C = c_j) = \sum_{s^i \in \Sigma^{|s|}} S(D, s^i, C = c_j)$. Hence, the queries of form $S(D, |s|, C = c_j)$(where$|s| = k$) needed in equation (4.2) need not be posed and can be computed from the queries of the form $S(D, s, C = c_j)$. With this optimization $QC(MM(k-1) = |C||\Sigma|^k$. However, the query complexity is still exponential in k. An approach to ameliorate this exponential explosion in the number of queries is to use the *lazy approach* to classify sequences, where instead of precomputing the results for all the possible queries ahead of time, only the answers to the queries needed to classify a given dataset of sequences are retrieved. Consider a test dataset $\mathcal{T} = \{s^1, s^2, \ldots s^t\}$ of t sequences that need to be classified. Given a sequence s, let $\Lambda(s, k)$ be the set of unique subsequences of length k in s. From equation (4.1) it follows that the required queries to classify a sequence s is $|C|(1 + |\Lambda(s, k)|)$. Hence, the query complexity of the lazy approach to classify the dataset $\mathcal{T}$ is $|C|(1 + \sum_{i=1}^{|\mathcal{T}|} |\Lambda(s^i, k)|)$. Since $\Lambda(s, k)$ is atmost $|s| - k + 1$ (i.e. when all subsequences of length k in s are unique), the query complexity $QC(MM(k-1)) \leq |C| \left(1 + \sum_{i=1}^{|\mathcal{T}|} (|s^i| - k + 1)\right)$.

The query complexity $QC(MM(k-1))$ can be further reduced through the use of *caching* wherein the system maintains a cache of answers to primitive queries answered during the execution of L against D. Before querying the data source D we can check if the answer to the query is available in the cache. Assuming that the sequences to be classified arrive in the order $s^1, s^2 \ldots s^t$, let $cache_i$ contain the answers to queries answered by D in the course of classifying sequences s^1 through s^{i-1}. Because the cache is initially empty, $cache_1 = \phi$. Let $\Lambda(s^i, k, D)$ denote the results to the queries for the counts, as obtained from D, of corresponding sequences in $\Lambda(s^i, k)$. Hence, $cache_{i+1} = \{\Lambda(s^1, k, D) \cup \Lambda(s^2, k, D) \ldots \cup \Lambda(s^i, k), D\}$. The additional queries needed to be posed to the data source D to classify sequence s^i given that the sequences $s^1 \ldots s^{i-1}$ have been already classified correspond to obtaining the set of counts $\Delta(s^i, k, D) = \Lambda(s^i, k, D) - \{\Lambda(s^1, k, D) \cup \Lambda(s^2, k, D) \ldots \cup \Lambda(s^{i-1}, k, D)\}$. Hence, the query complexity of the resulting approach is $QC(MM(k-1)) = |C|(1 + \Sigma_{i=1}^{|\mathcal{T}|}|\Lambda(s^i, k)|) - \{|\Lambda(s^{i-1}, k) \cup \Lambda(s^{i-2}, k) \ldots \Lambda(s^0, k)|\}$ with $\Lambda(s^0, k) = \phi$. **Observation**: The lazy approach for sequence classification is equivalent to the Markov Model based approach to classification that has access to the underlying dataset D. This follows from the fact that both approaches use equation (4.1) to compute the probability of a given sequence belonging to the class and this probability is the same when computed in the traditional way or through the use of statistical queries as in equation (4.2).

4.3.2 Interpolated Markov Models

Higher order Markov Models have a greater expressive power than their lower order counterparts. However, the higher the order of the Markov model, the less reliable are the estimates of the model parameters. The Interpolated Markov Models provide a means of dealing with this problem using a weighted combination of Markov models with several different choices of k (see Zhu et al. (2006a), Salzberg et al. (1998)). Given a sequence $s = \sigma_1\sigma_2 \ldots \sigma_n$ let $s_i = \sigma_1\sigma_2 \ldots \sigma_i$ be the subsequence ending at position i and $s_{i,j} = \sigma_{i-j}\sigma_{i-j+1} \ldots \sigma_{i-1}$ be sequence composed of the j positions that precede σ_i. Then the estimate of the probability of a sequence s belonging to the class c_j using an Interpolated Markov Model of order k is denoted by $\hat{P}_{IMM(k)}(s, c_j)$ and

$$\hat{P}_{IMM(k)}(s, c_j) = \Sigma_{i=1}^{n} IMM_k(s_i, c_j)$$

where $IMM_k(s_i, c_j) = \lambda_k(s_{i-1})\hat{P}_{MM(k)}(s_i, c_j) + (1 - \lambda_k(s_{i-1}))IMM_{k-1}(s_i, c_j)$ and $\lambda_k(s_{i-1})$ is the numeric weight associated with the k-mer ending at position $i-1$ in sequence s (i.e. $s_{i,k}$) and

$\hat{P}_{MM(k)}(s_i, c_j)$ is the estimate obtained from training data with the k^{th} order Markov model (see Salzberg et al. (1998), Salzberg et al. (1999) for details). The estimate $\hat{P}_{MM(k)}(s_i, c_j)$ required to build the Interpolated Markov Models (IMMs) can be computed, in the sufficient statistics model, as described earlier in section 4.3. Hence we need a way to compute the numeric weight $\lambda_k(s_{i-1})$ using only statistical queries. Consider for example, the computation of $\lambda_k(s_{i-1})$ in Glimmer [Salzberg et al. (1999)]. These weights can be computed in our setting using statistical queries of the form $S(D, s_{i,k}, C = c_j)$ and $S(D, s_{i,k}\sigma, C = c_j)$ where $\sigma \in \Sigma$. Specifically, $\lambda_k(s_{i-1}) = 1$ when $S(D, s_{i,k}, C = c_j)$ is greater than some threshold (for Glimmer the threshold is 400). When the count is less than the threshold, we compare the observed frequencies of $S(D, s_{i,k}\sigma, C = c_j)$ ($\sigma \in \Sigma$) with those predicted by IMM of order $k - 1$. Using a statistical test we compute the confidence (say d) that the observed frequencies are not consistent with those predicted by $\hat{P}_{IMM(k-1)}(s_{i,k}\sigma, c_j)$. When $d < 0.5$, $\lambda_k(s_{i-1}) = 0$ and for $d \geq 0.5$, $\lambda_k(s_{i-1}) = d/400 \times S(D, s_{i,k}, C = c_j)$. Thus Interpolated Markov Models can be implemented using statistical queries against the data source D.

4.4 Probabilistic Suffix Trees

The Probabilistic Suffix Trees (PSTs) originally introduced by Ron et al. [Ron et al. (1996)] have been successfully used to model and predict protein families [Bejerano and Yona (2001), Sun and Deogun (2004)]. The PSTs exploit the so called *short memory* feature of natural sequences wherein the probability distribution of the next symbol given the preceding sequence can be approximated by observing at most L preceding symbols of the sequence (L being the memory length of the PST). To use PSTs for sequence classification, we need to train a PST for each class; To classify an unlabeled sequence, we compute the probability of the sequence given the class (i.e., the corresponding PST) and assign it to the class with the largest probability. We first describe the algorithm to build a PST (say for class label $C = c_j$) using an available training dataset. The specific construction algorithm, **Build-PST**, is adapted from Bejerano and Yona (2001) and is described below. The procedure uses five external parameters: L the memory length, P_{min} the minimum probability which subsequences are required to occur and three parameters α, γ_{min} and r with values between zero and one (refer Bejerano and Yona (2001) for details). The procedure uses $\bar{T}$ to denote the PST and $\bar{T}$ is constructed iteratively starting with the root node. Each node (say labeled with s) maintains a vector $\bar{\gamma}_s$ which encodes the probability distribution (over the next symbol) associated with the node s (we use $\bar{\gamma}_s(\sigma)$ to denote the probability of the symbol σ in the distribution $\bar{\gamma}_s$).

Algorithm: **Build-PST**$(P_{min}, \alpha, \gamma_{min}, r, L)$

(1) *Initialization*: let $\bar{T}$ consists of a single root node (with an empty label), and let $\bar{S} \leftarrow \{\sigma | \sigma \in \Sigma \text{ and } P\hat{(\sigma)} \leq P_{min}\}$

(2) *Building the PST skeleton*: while $\bar{S} \neq \phi$, pick any $s \in \bar{S}$ and do:

(a) Remove s from $\bar{S}$

(b) If there exists a symbol $\sigma \in \Sigma$ such that

$$\hat{P}(\sigma|s) \geq (1+\alpha)\gamma_{min}$$

and

$$\frac{\hat{P}(\sigma|s)}{\hat{P}(\sigma|suffix(s))} \begin{cases} \geq r \\ or \\ \leq 1/r \end{cases}$$

then add to $\bar{T}$ the node corresponding to s and all the nodes on the path to s from the deepest node in $\bar{T}$ that is a suffix of s.

(c) If $|s| \leq L$ then add the strings $\{\acute{\sigma}s | \acute{\sigma} \in \Sigma \text{ and } \hat{P}(\acute{\sigma}s) \geq P_{min}\}$ (if any) to $\bar{S}$.

(3) *Smoothing the prediction probabilities* For each s labeling a node in $\bar{T}$, let

$$\bar{\gamma}_s(\sigma) = (1 - |\Sigma|\gamma_{min})\hat{P}(\sigma|s) + \gamma_{min}$$

Note the final step (step(3)) of the algorithm corresponds to a parameter smoothing step.

Build-PST iteratively adds nodes (step (2)) to obtain a PST. The terms calculated in step (2) are $\hat{P}(\sigma|s)$, $\hat{P}(\acute{\sigma}s)$ and $\hat{P}(\sigma|suffix(s))$. These terms can be calculated using statistical queries as follows:

- $\hat{P}(\sigma|s) = \frac{S(D,s\sigma,C=c_j)}{S(D,|s\sigma|,C=c_j)}$

- $\hat{P}(\acute{\sigma}|s) = \frac{S(D,\acute{\sigma}s,C=c_j)}{S(D,|\acute{\sigma}s|,C=c_j)}$

- $\hat{P}(\sigma|suffix(s)) = \frac{S(D,suffix(s)\sigma,C=c_j)}{S(D,|suffix(s)\sigma|,C=c_j)}$

Since $|s\sigma| = |\acute{\sigma}s|$, it follows that the query $S(D, |s\sigma|, C = c_j)$ is the same as $S(D, |\acute{\sigma}s|, C = c_j)$. Hence, in each iteration of step (2), requires five different queries to be answered by D. If $r(D, c_i)$ is the number of times the step (2) is executed during the construction of the PST for class $c_i \in C$, then $QC(PST) = 5\sum_{i=1}^{|C|} r(D, c_i)$. In practice $r(D, c_i)$ and hence the Query Complexity depends on the dataset D as well as choice of P_{min}. However, in the worst case the number of queries submitted is bounded by the number of queries needed to to build Markov Models of length through 1 and L. Hence, the query complexity $QC(PST) \leq |C|\sum_{k=1}^{L} |\Sigma|^k$.

4.5 Updatable Predictive Models

The advent of automated high throughput sequencing techniques has resulted in an exponential increase in the rate at which genomic sequence data is being generated. Many practical applications call for techniques that allow the predictive models to be updated without the need to regenerate the model from scratch. The update can either be *additive* wherein new data needs to be incorporated into the model or *subtractive* wherein the contributions of some of the old data need to be discarded from the model.

Given a dataset D and a learning algorithm ψ, let $\psi(D)$ be a predictive model (e.g., a Markov model) built from the data set D using a learning algorithm ψ. In the sufficient statistics model, let $\theta^{\psi}(D)$ be the set of primitive queries required over dataset D to build $\psi(D)$. In order for the built model ψ to incorporate new data (without rebuilding the model from scratch) it should be able to able to build a model over the combined dataset using additional statistical queries posed only over the new data. A similar argument holds for removing a subset of the data from the model. We formalize this notion in terms of an *Updatable Model.*

Definition 1 ***Updatable Model*** *: Given datasets D_1 and D_2 such that $D_1 \subseteq D_2$, we say that the predictive model constructed using ψ is updatable iff we can specify functions f and g such that*

1. $\theta^{\psi}(D_2) = f(\theta^{\psi}(D_2 - D_1), \theta^{\psi}(D_1))$

2. $\theta^{\psi}(D_1) = g(\theta^{\psi}(D_2), \theta^{\psi}(D_2 - D_1))$

Theorem 1 *Markov Models are* updatable *by a statistical query based learning algorithm.*

Proof 1 *For a Markov Model queries of the form $S(D, s, C = c_j)$ and $S(D, |s|, C = c_j)$ over $Desc^s(D)$ form the set $\theta^{\psi}(D)$ (see equations (4.1) and (4.2)). Given datasets D_2 and D_1 such that $D_1 \subseteq D_2$, it is easy to see that $S(D_2, s, C = c_j) = S(D_2 - D_1, s, C = c_j) + S(D_1, s, C = c_j)$. Similarly, $S(D_2, |s|, C = c_j) = S(D_2 - D_1, |s|, C = c_j) + S(D_1, |s|, C = c_j)$. As a result the set $\theta^{\psi}(D_2)$ can be constructed from $\theta^{\psi}(D_2 - D_1)$ and $\theta^{\psi}(D_1)$. Similarly, $S(D_1, s, C = c_k) = S(D_2, s, C = c_k) - S(D_2 - D_1, s, C = c_k)$ and $S(D_1, |s|, C = c_k) = S(D_2, |s|, C = c_k) - S(D_2 - D_1, |s|, C = c_k)$. Consequently the set $\theta^{\psi}(D_1)$ can be constructed from $\theta^{\psi}(D_2)$ and $\theta^{\psi}(D_2 - D_1)$.*

The theorem above shows that the Markov Model is updatable in the statistical query model and hence provides a natural way to incorporate new data or remove unwanted data in setting of building Markov Model using statistical queries.

Observation: The PST built using the **Build-PST** procedure is not updatable. This is due to the fact that in step 2(c) a string is added to set $\bar{S}$ only if it probability is greater than P_{min}. It is possible that this condition is satisfied for a string (say x) in $D_2 - D_1$ but not in D_1 (say when x never occurs in D_1 but occurs in $D_2 - D_1$). As a result the queries to estimate $P(\sigma|x)$ from dataset D_1 are never posed while being posed for the dataset $D_2 - D_1$. Hence, the PST is not updatable since it is not possible to compute $\hat{P}(\sigma|x)$ for the dataset D_2 only from queries posed to compute $\hat{P}(\sigma|x)$ for dataset $D_2 - D_1$

4.6 Discussion

In the preceding sections, we focused on the Markov property based classifiers that can learn from sequence data using only count queries of the form $S(D, s, C = c_k)$. However, one of the important classifiers in this family is the Hidden Markov Model based classifier (see Rabiner (1990) for an overview). Hidden Markov Model (HMM) based approaches have been used to address a variety of learning tasks from sequence data [Rabiner (1990), Stanke and Waack (2003), Martin et al. (2005), Zhu et al. (2006a)]. We do not address the HMM based approach since the implementation of Baum-Welch algorithm in the statistical query approach is not straightforward. We beleieve it will involve certain simplifying assumptions (say fixing the number of hidden states) and additional type of queries besides $S(D, s, C = c_k)$ (say in the Gene finding approach, the queries to compute the transition matrix between the states).

In our setting of computing the Markov Property based class of predictors using statistical queries we assumed that the data source holding the dataset answers queries of the form $S(D, s, C = c_k)$. In practice the sequence data is often accessible via the web and the data source provider often provides a variety of tools to access the data and it is not difficult envision support for such type of queries. However, we show that supporting these types of queries is fairly straightforward for a dataset stored in an RDBMS. Note that we acknowledge existing commercial RDBMS may not be an efficient choice for efficiently storing and querying sequence data [Stonebraker (2005), Stein (2010)] . In fact, there is a pressing need for storage solutions to the massive amount of sequence data that is being generated at an enormous rate. An approach to address a related problem (Web indexing) that generates Petabytes amount of text data is BigTable [Chang et al. (2006)]. However, assuming that dataset is stored in a RDBMS the queries of the form $S(D, s, C = c_k)$ can be easily supported. Given $Desc^s(D)$, let us assume the data is stored in a RDBMS with the following schema: the data is stored in a table named D with the attributes *id*, *sequenceData*, and *class* where *id* is a unique identifier of the sequence (primary key), the attribute *sequenceData* contains the representation of the sequence as a string and the attribute *class* specifies the family to which the particular sequence belongs (note that it is fairly straightforward to dump FASTA or XML formatted data into this schema). In such a setting, the support for queries of the format $S(D, s, C = c_j)$ and $S(D, |s|, C = c_j)$ can be added by the user by writing appropriate constructs using a procedural language (say PL/SQL) supported by the database. However, in general the data sources holding D may not allow user written code to be executed on their system. In such a case, the support for these queries needs to be provided in terms of standard SQL queries. We propose the use of the regular expression support in SQL queries provided by RDMS. Let $R(D, C = c_j, expr)$ represent the SQL query corresponding to the count of the number of instances in D that have the value of the attribute class as c_j and the sequenceData matches the regular expression *expr*. The exact syntax of $R(D, C = c_j, expr)$ will depend on the particular RDBMS under use. For Oracle 10g database, the $R(D, C = c_j, expr)$ will be the SQL statement *Select count(id) from D where REGEX_LIKE (sequence expr) AND class* $= c_j$. Similarly for MySQL $R(D, C = c_j, expr)$ be the SQL statement *Select count(id) from D where sequenceData REGEX expr AND class* $= c_j$. Consider the regular expression $\hat{}[.]\{n\}s$ which matches any n characters followed by the string s. Assuming l_{max} is the maximum possible length of a sequence, we have

$$S(D, s, C = c_j) = \sum_{n \in \{0,1 \ldots l_{max}\}} R(D, C = c_j, \hat{}[.]\{n\}s)$$

Similarly, the regular expression$\hat{}[.]\{n+k\}$ matches any n+k characters starting from the beginning of the line. Then,

$$S(D, |s|, C = c_j) = \sum_{n \in \{0,1 \ldots l_{max}\}} R(D, C = c_j, \hat{}[.]\{n+k\})$$

As a result to calculate a single query of the form $S(D, s, C = c_j)$ (or $S(D, |s|, C = c_j)$, l_{max} number of SQL queries are submitted to the database. Hence, the computed Query Complexity is increased by a factor of l_{max}. However, the preceding discussion demonstrates that it is possible to provide support for the type of queries required to build Markov Models using only statistical queries.

Observation: It is possible to compute the answer to the query from answers to the queries of the form $S(D, |s|, C = c_j)$ as $S(D, |s|, C = c_j) = \sum_{x \in \Sigma^{|s|}} S(D, x, C = c_j)$. However, it requires $|\Sigma^{|s|}|$ queries as opposed to l_{max} queries using the approach suggested earlier and may be useful only in cases when results to queries of the form $S(D, s, C = c_j)$ are already available (say in cache).

4.7 Summary and Related Work

Summary: Due to the exponential increase in the rate at which sequence data are being generated, there is an urgent need for efficient algorithms for learning predictive models of sequence data from large sequence databases and for updating the learned models to accommodate additions or deletions of data in settings where the sequence database can answer only a certain class of statistical queries.

In this chapter we presented an approach to learning predictive models from sequence data using sufficient statistics by posing count queries against a sequence data source. This approach can be used to build the predictive model without access to the underlying data as long the data source is able to answer a class of count queries. In addition, this approach scales well to settings where the dataset is very large in size because it does not need to load the entire dataset in memory. We have also outlined some optimization techniques to minimize the number of queries submitted to the data source. In addition, we showed how the class of Markov model based predictors can be updated in response to addition or deletion of subsets of the data.

Related Work: The approach to learning Markov models and their variants presented in this chapter builds on the statistical query based approach to learning from large datasets (including distributed data sets) introduced by Caragea et al. (2004b). Markov Models have been successfully used in a broad range of applications in computational biology including gene finding (e.g. GeneMark [Borodovsky and McIninch (1993)], GenScan [Burge and Karlin (1997)]), protein classification [Yuan (1999), Yakhnenko et al. (2005), Fischer et al. (2004)] , among others. Salzberg et al. (1998) have used Interpolated Markov Models for gene finding. Bejerano and Yona (2001) and Sun and Deogun (2004) have used Probabilistic Suffix Trees for protein classification. Begleiter et al. (2004) discuss Variable order Markov Models. Abouelhoda et al. (2004) have investigated approaches to reducing the memory requirements of suffix tree construction algorithms. Koul et al. (2008) describe approaches to build Naive Bayes and Decision Trees from databases using SQL queries.

CHAPTER 5. Learning In Presence of Ontology Mapping Errors

The widespread use of ontologies to associate semantics with data has resulted in a growing interest in the problem of learning predictive models from data sources that use different ontologies to model the same underlying domain (world of interest). Learning from such *semantically disparate* data sources involves the use of mapping to resolve semantic disparity among the ontologies used. Often, in practice, the mapping used to resolve the semantic disparity may contain errors and as such the learning algorithms used in such a setting must be robust in presence of mapping errors. We reduce the problem of learning from semantically disparate data sources in the presence of mapping errors to the problem of learning in the presence of *nasty* classification noise. This reduction allows us to transfer theoretical results and algorithms from the latter to the former.

5.1 Introduction

Recent advances in high throughput data acquisition technologies in many applications have resulted in a proliferation of autonomous and distributed data sources. Different data sources often use disparate vocabularies (e.g., M.S. student versus Masters student), units (e.g., temperature measured in degrees Centigrade versus Fahrenheit), and levels of detail (e.g. graduate student, student) to describe the objects of interest in the world being modeled. In such a setting, different data sources represent different conceptual models of the same underlying world. In the semantic web vision this typically translates to each data source assuming a particular ontology to model objects, properties and relationships in the world of interest.

Hence, learning from such data sources requires reconciling the semantic differences between the learner's conceptual model of the world (i.e., learner's ontology) and the models of the world associated with the the disparate data sources (i.e., the data source ontologies). This is achieved through a data integration step [Lenzerini (2002), Hull (1997)] that presents to the learning algorithm, a single *view* of the different data sources. Data integration involves mapping the terms in the data source ontologies to the corresponding terms in the learner's ontology (see Kalfoglou and Schorlemmer (2005) for a survey). However, this mapping process is often error prone. Errors in mappings can be due to human error, errors in the automated mapping algorithm used, or by lack of exact correspondences between terms in a source ontology and the target ontology. Hence, it is of interest to characterize the effect of mapping errors on the accuracy of the predictive models (e.g., classifiers) learned in such a setting.

Consider for example, the problem of *learning boolean conjunctions.* The target function $f : \{0,1\}^n \longrightarrow \{0,1\}$ to be learned is a conjunction of literals drawn from $x_1, x_2 \ldots x_n$ and their negations. The learner L expects the training examples of the form $(x,\ y)$ where $x \in \{0,1\}^n$ and $y \in \{0,1\}$. Suppose the learner receives the training data (labeled examples) from two different sources D_1 and D_2. Let each instances from D_1 (as well as D_2) correspond to an n valued attribute value vector, and an ontology associated with each attribute (and the label) specify the possible values that the attribute can take. Let the ontology associated with each attribute and the class label in D_1 be such that each attribute as well as the class label can take either then boolean value *True* or *False.* Hence, the data from D_1 is of the form $(x,\ y)$ where $x \in \{True, False\}^n$ and $y \in \{True, False\}$. Similarly, let the ontologies associated with the attributes and the class label in D_2 be such that the data from D_2 is of the form $(x,\ y)$ where $x \in \{-5V, +5V\}^n$ and $y \in \{Off, On\}$ (Note data in D_2 can be seen as record of the input voltages, and output to boolean circuit that encodes the function f). Now in order for the learner to be able to learn from data from both D_1 and D_2, the vocabularies used by D_1 and D_2 have to be mapped to the vocabulary used by the learner. Thus, suppose -5V, +5V, *Off*, *On* in D_2 map to 0, 1, 0, 1 (respectively) from the learner's point of view. Similarly, $False$ and $True$ in D_1 *map to 0 and 1*(respectively) from the learner's point of view. These mappings transform examples from D_1 and D_2 into examples from which L can learn the unknown target function *f*. While in this example, the mappings were relatively simple, in real-world applications, establishing such mappings and ensuring that they preserve the *intended* semantics can be a complex, and error-prone process. Suppose in our example because of human error, the *On* is incorrectly mapped to *0* instead of 1. As a consequence of this mapping error, some of the instances from D_1 are incorrectly labeled from the learner's point of view.

Against this background, this paper reduces the problem of learning from semantically disparate data sources in the presence of mapping errors to a variant of the problem of learning in the presence of *nasty* classification noise in a PAC-like framework (see Valiant (1984), Kearns and Vazirani (1994) for background on PAC learning). This reduction allows us to transfer results and algorithms from latter to the former. This reduction proves to be very useful in practice as techniques to deal with noise have been well studied in literature and can be applied to the setting of learning in presence of mapping errors.

The rest of the paper is organized as follows. Section 5.2 introduces a formal model of learning from disparate data sources. Section 5.3 presents the main result of the paper: that learning from disparate data sources in the presence of mapping errors can be modeled by learning from a single data source in the presence of noise. Finally Section 6.5 concludes with a summary, significance, and a brief discussion of related work.

5.2 Learning from Semantically Disparate Data Sources

We now introduce the notion of a *k-Delegating Oracle* to model learning from multiple data sources. We then extend the model to a *mapping aware k-Delegating Oracle* to model learning from semantically disparate data sources.

5.2.1 *k*-Delegating Oracle

Let $\mathcal{X}$ be an instance space, $\mathcal{D}$ a probability distribution over $\mathcal{X}$, $\mathcal{F}$ a function space and $f : \mathcal{X} \longrightarrow \{0, 1\}$ the target function to be learned ($f \in \mathcal{F}$). An oracle $EX(f, \mathcal{X}, \mathcal{D})$ is a procedure that returns a labeled example $\langle x, f(x) \rangle$ where x is drawn from $\mathcal{X}$ according to $\mathcal{D}$.

We use the notation $Pr_{x \in \mathcal{D}}[x]$ to indicate the probability of drawing an instance x from $\mathcal{X}$ according to the distribution $\mathcal{D}$. The classical model of supervised learning, consisting of a learner L with access to an Oracle $EX(f, \mathcal{X}, \mathcal{D})$, is not expressive enough to model learning from multiple data sources. Consequently we introduce the notion of a *k-Delegating Oracle* to model learning from multiple data sources.

A *k-delegating oracle* $kEX(f, \mathcal{X}, \mathcal{D})$ invokes subordinate oracles $EX^1(f, \mathcal{X}, \mathcal{D}_1), \ldots EX^k(f, \mathcal{X}, \mathcal{D}_k)$ with probabilities $p_1 \ldots p_k$ respectively. The i^{th} oracle $EX^i(f, \mathcal{X}, \mathcal{D}_i)$ when invoked returns an example of the form $\langle x, f(x) \rangle$ where x is drawn from $\mathcal{X}$ according to $\mathcal{D}_i$. The distribution $\mathcal{D}$ of the k-delegating oracle is $Pr_{x \in \mathcal{D}}[x] = \sum_{i=1}^{k} p_i \times Pr_{x \in \mathcal{D}_i}[x]$

A *classical oracle* can be seen as a special case of the k-delegating oracle with $k = 1$.

5.2.2 Mapping Aware k-Delegating Oracle

Let $\mathcal{X}_{s^1}, \mathcal{X}_{s^2} \ldots \mathcal{X}_{s^k}$ be k instances spaces; Let $\mathcal{D}_1, \mathcal{D}_2 \ldots \mathcal{D}_k$ be probability distributions over $\mathcal{X}_{s^1}, \mathcal{X}_{s^2} \ldots \mathcal{X}_{s^k}$ respectively and $\mathcal{F}^1, \mathcal{F}^2 \ldots \mathcal{F}^k$ be k functions spaces defined over the corresponding instance spaces where each function in $\mathcal{F}^i$ labels instances in $\mathcal{X}_{s^i}$ with a label in the set C_i.

A *mapping aware k-delegating oracle* has access to a mapping set $M = \{m_1, m_2 \ldots m_k\}$ where $m_i = \{m_i^x, m_i^c\}$; $m_i^x : \mathcal{X}_{s^i} \longrightarrow \mathcal{X}$ is an *attribute mapping function*; and $m_i^c : C_i \longrightarrow C$ is a *class mapping function* where $C_i = Range(f_i)$ and $C = Range(f)$. It invokes subordinate oracles $EX^1(f_1, \mathcal{X}_{s^1}, \mathcal{D}_1) \ldots EX^k(f_k, \mathcal{X}_{s^k}, \mathcal{D}_k)$ where the i^{th} subordinate oracle $EX^i(f_i, \mathcal{X}_{s^i}, \mathcal{D}_i)$ returns examples of the form $\langle x_{s^i}, f_i(x_{s^i}) \rangle$ where x_{s^i} is drawn from X_{s^i} according to $\mathcal{D}_i$ and $f_i \in \mathcal{F}^i$. It uses the mapping m_i to convert an instance $\langle x_{s^i}, f_i(x_{s^i}) \rangle$ received form the i^{th} subordinate oracle to $\langle m_i^x(x_{s^i}), m_i^c(f_i(x_{s^i})) \rangle$ before passing it to the learner. We assume the mappings m_i are computable and satisfy the following conditions: $\forall x_{s^i} \in \mathcal{X}_{s^i}, m_i^x(x_{s^i}) \in \mathcal{X}$; $\forall l \in C_i, m_i^c(l) \in C$ and whenever $x \in \mathcal{X}_{s^i}, \mathcal{X}_{s^j}$, $m_i^x(x) = m_j^x(x)$. These conditions ensure that the examples returned by the mapping aware k-delegating oracle are of the form $\langle x, l(x) \rangle$ where $x \in \mathcal{X}$ and $l(x) \in C$.

Ideally the mappings should ensure that the examples returned to the learner are labeled

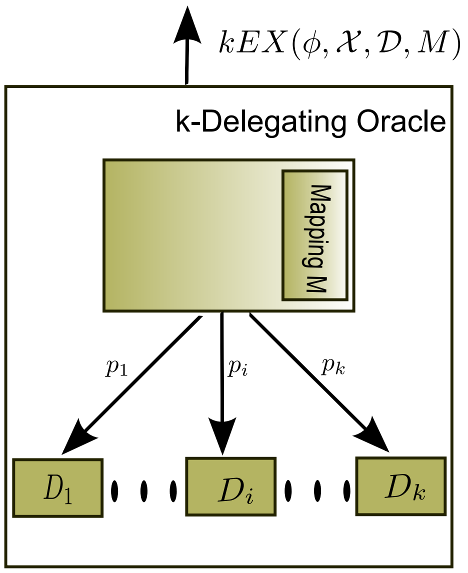

Figure 5.1 A schematic representation of mapping aware k-delegating oracle.

according to the target function f. However, in practice mappings may have errors and consequently the instances may labeled according to ϕ which may be different from f. We denote the mapping aware k-delegating oracle by $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ where ϕ is the labeling function. A schematic representation of mapping aware k-delegating oracle is shown in Figure 5.1.

Let $Y^x_{s^i}$ be a set that consists of all the elements in $\mathcal{X}_{s^i}$ that are mapped to an element $x \in \mathcal{X}$ using the mapping m^x_i. Then the distribution $\mathcal{D}$ over $\mathcal{X}$ is given by

$$Pr_{x \in \mathcal{D}}[x] = \sum_{i=1}^{k} \sum_{y \in Y^x_{s^i}} p_i \times Pr_{y \in \mathcal{D}_i}[y] \tag{5.1}$$

Note that the sampling distribution $\mathcal{D}$ now depends on mappings $m^x_1 \ldots m^x_k$ (because of dependence on $Y^x_{s^i}$).

5.3 Learning in the Presence of Mapping Errors

We now proceed to describe (formally) what it means for a mapping to be correct (and correspondingly to have errors) and establish an equivalence between learning in the presence of mapping errors and learning from noisy data.

5.3.1 Mapping Errors

The sets of class labels $C_1 \ldots C_k$ as well C partition the corresponding instance spaces $\mathcal{X}_{s_1} \ldots \mathcal{X}_{s_k}$ and $\mathcal{X}$ respectively. Each cell in a partition corresponds to a set of instances that share the same class label. The mapping m_i^c establishes a correspondence between the cells of the partition of $\mathcal{X}_{s_i}$ and those of the partition of $\mathcal{X}$. We define errors in mappings relative to a reference set of mappings $m^c_{1,expert}(l) \ldots m^c_{k,expert}(l)$ (e.g., provided by an expert).

Definition 4 (Correct Class Mapping) *A class mapping m_i^c is said to be correct if $\forall l \in C_i$ $m_i^c(l) = m^c_{i,expert}(l)$.*

Definition 5 (Correct Attribute Mapping) *An attribute mapping m_i^x is said to be correct whenever $\forall x \in \mathcal{X}_{s^i}$, $f_i(x) = l$ and $m^c_{i,expert}(l) = l_1 \longrightarrow f(m_i^x(x)) = l_1$*

Definition 6 (Correct Mapping Set) *A mapping set $M = \{m_1, m_2 \ldots m_k\}$ is said to be correct if $\forall i \in \{1, 2, \ldots, k\}$ m_i^x and m_i^c are correct.*

In the rest of the paper, we assume that a correct class label mapping is available (say from a domain expert) and all mapping errors are attribute mapping errors. Figure 5.2 and Figure 5.3 show an example of a correct mapping and an incorrect mapping respectively. The following observation follows directly from the above definitions.

Observation 1: Given a correct attribute mapping m_i^x it follows that $m^c_{i,expert}(f_i(x_{s^i})) = f(m_i^x(x_{s^i}))$

Suppose the k-delegating oracle invokes the i^{th} subordinate oracle. Then the labeled example passed to the learner is of the form $\langle m_i^x(x_{s^i}), m_i^c(f_i(x_{s^i}))\rangle$ where $x_{s^i} \in X_{s^i}$. Because m_i^c is assumed to be the same as $m^c_{i,expert}$, it follows from Observation 1 that $\langle m_i^x(x_{s^i}), m_i^c(f_i(x_{s^i}))\rangle$

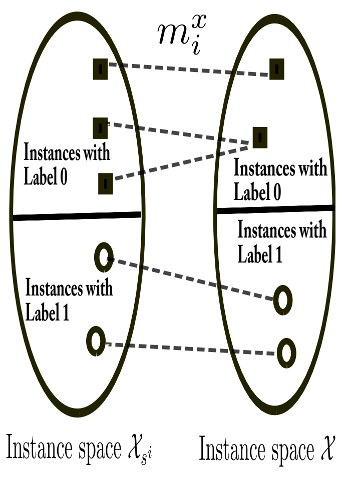

Figure 5.2 An example of a correct mapping.

$= \langle m_i^x(x_{s^i}), f(m_i^x(x_{s^i}))\rangle = \langle x, f(x)\rangle$ where $x \in \mathcal{X}$ since $m_i^x(x_{s^i}) \in \mathcal{X}$. Hence we have the following observation.

Observation 2: Given a correct mappings set M, for each labeled example of the form $\langle x, \phi(x)\rangle \in \mathcal{X} \times C$ provided by $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$, it must be the case that $\phi(x) = f(x)$ where f is the target function.

Observation 2 shows that when the mappings have no errors the instances passed to the learner are labeled according to the target function f.

5.3.2 Mapping Errors as Noise

We now proceed to show that the mapping errors result in the incorrectly labeled examples being provided to the learner and hence can be seen as introducing classification noise in the examples. Let $Pr_{x\in\mathcal{D}}[e = \langle x, f(x)\rangle]$ denote the probability that a labeled example $e = \langle x, f(x)\rangle$ is obtained by a single call to the oracle $EX(f, \mathcal{X}, \mathcal{D})$.

Definition 7 (Equivalent Oracles) *The oracles* $EX1(f_1, \mathcal{X}, \mathcal{D}_1)$ *and* $EX2(f_2, \mathcal{X}, \mathcal{D}_2)$ *are said to be equivalent whenever* $\forall e \in \mathcal{X} \times Range(f_1) \cup Range(f_2), Pr_{x\in\mathcal{D}_1}[e = \langle x, f_1(x)\rangle] = Pr_{x\in\mathcal{D}_2}[e = \langle x, f_2(x)\rangle]$

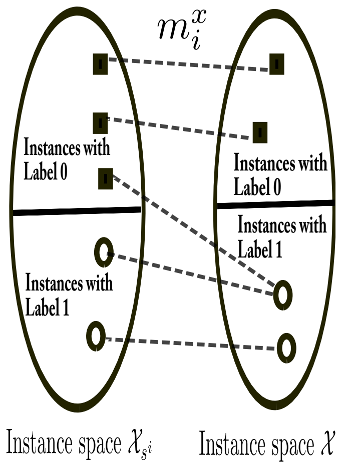

Figure 5.3 An example of a mapping with errors.

The following observation follows directly from Observation 2 and the definition of equivalent oracles.

Observation 3: A k-delegating oracle $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ is equivalent to a classical oracle $EX(f, \mathcal{X}, \mathcal{D})$ whenever the mapping set M is correct with respect to target function f.

Definition 8 (Noisy Oracle) *Let* $\eta_x : \mathcal{X} \mapsto [0, 1]$ *be an* instance dependent classification noise rate. *A noisy oracle* $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$ *operates as follows: It calls a classical oracle* $EX(f, \mathcal{X}, \mathcal{D}_{eq})$ *to obtain a labeled example* $\langle x, f(x) \rangle$ *and returns to the learner the example* $\langle x, f(x) \rangle$ *with a probability* $1 - \eta_x$ *and* $\langle x, 1 - f(x) \rangle$ *with probability* η_x.

Given a k-delegating oracle $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$, let $\beta(x)$ be the probability that an instance x obtained by a single call to $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ has the label $\phi(x)$ which is different from $f(x)$. Let $\gamma(x)$ be the probability that an instance x obtained by a single call to $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ has the label $\phi(x)$ which is same as $f(x)$.

Theorem 2 *A* k-*delegating oracle* $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ *is equivalent to a noisy oracle* $EX1_{\eta_x}$ $(f, \mathcal{X}, \mathcal{D}_{eq})$ *when the distributions* $\mathcal{D}$ *and* $\mathcal{D}_{eq}$ *are identical and* $\eta_x = \frac{\beta(x)}{\beta(x)+\gamma(x)}$

Proof 2 *From (5.1), the distribution $\mathcal{D}$ over $\mathcal{X}$ of the given* k*-delegating oracle is*

$$Pr_{x\in\mathcal{D}}[x] = \sum_{i=1}^{k}\sum_{y\in Y^x_{s^i}} p_i \times Pr_{y\in\mathcal{D}_i}[y]$$

We define $\alpha_i(x) = \sum_{y\in Y^x_{s^i}} Pr_{y\in\mathcal{D}_i}[y]$, then

$$Pr_{x\in\mathcal{D}}[x] = \sum_{i=1}^{k} p_i \times \alpha_i(x)$$

Now $\alpha_i(x)$ can be seen as the weight (sum of probabilities) of instances drawn from X^i_s according to $\mathcal{D}_i$ that is mapped to $x \in \mathcal{X}$. In presence of mapping errors let the set $Y^x_{s^i} = A^x_{s^i} \cup B^x_{s^i}$ where $A^x_{s^i}$ is subset of instances in $Y^x_{s^i}$ that are correctly mapped to $x \in \mathcal{X}$ while $B^x_{s^i}$ is the subset of instances in $Y^x_{s^i}$ that get mapped to $x \in \mathcal{X}$ due to mapping errors. Let $\gamma_i(x) = \sum_{y\in A^x_{s^i}} Pr_{y\in\mathcal{X}_{s^i},\mathcal{D}_i}[y]$ and $\beta_i(x) = \sum_{y\in B^x_{s^i}} Pr_{y\in\mathcal{X}_{s^i},\mathcal{D}_i}[y]$. Note that $\beta_i(x)$ and $\gamma_i(x)$ (respectively) are the weights of instances drawn from X^i_s according to $\mathcal{D}_i$ that are incorrectly and correctly mapped to $x \in \mathcal{X}$. Recall that $x \in X^i_s$ is correctly mapped using m^x_i if the following holds

$$f_i(x) = l_1 \, and \, m^c_{i,expert}(l_1) = l \longrightarrow f(m^x_i(x)) = l.$$

It follows that

$$\alpha_i(x) = \beta_i(x) + \gamma_i(x)$$

In addition $\gamma(x) = \sum_{i=1}^{k} p_i \times \gamma_i(x)$ and $\beta(x) = \sum_{i=1}^{k} p_i \times \beta_i(x)$. Note that $\beta(x)$ is the probability that given the instance x is drawn (from $\mathcal{X}$ according to $\mathcal{D}$), it is labeled incorrectly . Similarly $\gamma(x)$ is the probability that given the instance x is drawn (again from $\mathcal{X}$ according to $\mathcal{D}$), it is labeled correctly. Hence

$$Pr_{x\in\mathcal{D}}[x] = \gamma(x) + \beta(x)$$

To avoid cluttering the notation we will abbreviate $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ and $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$ by kEX and $EX1_{\eta_x}$ respectively when the parameters are obvious from the context. Consider a labeled example $e = \langle x, l(x)\rangle \in E = \mathcal{X} \times \{0, 1\}$ where $l(x)$ is the label associated with x. The labeled example $e = \langle x, l(x)\rangle$ can be sampled either from kEX or $EX1_{\eta_x}$ and since the class *labels are binary $l(x)$ is either $f(x)$ of $1 - f(x)$.*

case : $l(x) = f(x)$

$$Pr_{x \in \mathcal{D}_{eq}}[e = \langle x, f(x) \rangle] = (1 - \eta_x) Pr_{x \in \mathcal{D}_{eq}}[x]$$

$$Pr_{x \in \mathcal{D}}[e = \langle x, f(x) \rangle] = \gamma(x)$$

case: $l(x) = 1 - f(x)$

$$Pr_{x \in \mathcal{D}_{eq}}[e = \langle x, 1 - f(x) \rangle] = \eta_x Pr_{x \in \mathcal{D}_{eq}}[x]$$

$$Pr_{x \in \mathcal{D}}[e = \langle x, f(x) \rangle] = \beta(x)$$

When the noisy oracle $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$ is such that

$$Pr_{x \in \mathcal{D}_{eq}}[x] = Pr_{x \in \mathcal{D}}[x] = \gamma(x) + \beta(x) = \sum_{i=1}^{k} p_i \times \alpha_i(x)$$

and

$$\eta_x = \frac{\beta(x)}{\beta(x) + \gamma(x)}$$

it follows that for either case

$$Pr_{x \in \mathcal{D}_{eq}}[e = \langle x, f(x) \rangle] = Pr_{x \in \mathcal{D}}[e = \langle x, f(x) \rangle] \tag{5.2}$$

This establishes the equivalence of oracles $EX1_{\eta_x}$ and kEX.

The theorem shows that the effect of the mapping errors in the k-delegating oracle kEX can be simulated by the noise function η_x associated with $EX1_{\eta_x}$.

5.3.3 Mapping Errors as Nasty Noise

We now argue that the noise model η_x associated with $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D})$ can be simulated by the nasty classification noise Bshouty et al. (1999) model which in turn can be simulated by the nasty sample noise model Bshouty et al. (1999).

Definition 9 ***(Instance Dependent Classification Noise(IDCN) Oracle)****: An Instance Dependent Classification Noise Oracle, denoted by $IDCN(m, \eta_x, f, \mathcal{X}, \mathcal{D})$, is one where an* *intermediary obtains a dataset D^m of m i.i.d examples by making m calls to a noisy oracle $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$. The resulting dataset is then provided to the learner.*

Definition 10 ***(k^m-delegating Oracle):*** *A k^m-delegating Oracle, denoted by $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$, is one where an intermediary obtains a dataset D^m of m i.i.d examples by making m calls to a* k*-delegating oracle $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$. The resulting dataset is then provided to the learner.*

Definition 11 ***(Nasty Sample Noise (NSN) Oracle (adapted from Bshouty et al. (1999))):*** *A Nasty Sample Noise Oracle, denoted by $NSN(m, \eta, f, \mathcal{X}, \mathcal{D})$, is one where an adversary obtains a dataset D^m of m i.i.d examples by making m calls to a classical oracle $EX(f, \mathcal{X}, \mathcal{D})$. The adversary then picks n out of m instances of its choosing from D^m (where n is distributed according to a binomial distribution with parameters m and nasty noise rate η) and replaces them with any examples of its choice from $\mathcal{X} \times Range(f)$. The resulting dataset is then provided to the learner.*

Definition 12 ***(Nasty Classification Noise (NCN) Oracle (adapted from Bshouty et al. (1999))):*** *A Nasty Classification Noise Oracle, denoted by $NCN(m, \eta, f, \mathcal{X}, \mathcal{D})$, is one where an adversary obtains a dataset D^m of m i.i.d examples by making m calls to a classical oracle $EX(f, \mathcal{X}, \mathcal{D})$. The adversary then picks n out of m instances of its choosing from D^m (where n is distributed according to a binomial distribution with parameters m and nasty noise rate η) and flips their class labels. The resulting dataset is then provided to the learner.*

Nasty Classification Noise (NCN) is a *weaker* case of NSN where the adversary is constrained such that it can modify only the class labels of n instances selected from D^m in a manner identical to that in the case of NSN

Consider a dataset obtained from $IDCN(m, \eta_x, f, \mathcal{X}, \mathcal{D})$. Let $\lambda = \sum_{x \in \mathcal{X}} \eta_x \times Pr_{x \in \mathcal{D}}[x]$. The value λ represents the probability that a random example in the dataset obtained by a single call to $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D})$ is mislabeled. The number of examples in the dataset obtained from $IDCN(m, \eta_x, f, \mathcal{X}, \mathcal{D})$ that have incorrect labels with respect to f can be viewed as number of successes in a sequence of m independent binary experiments each with a success probability λ. In the case of $NCN(m, \eta, f, \mathcal{X}, \mathcal{D})$, if we choose $\eta = \lambda = \sum_{x \in \mathcal{X}} \eta_x \times Pr_{x \in \mathcal{D}}[x]$, it follows that the number of incorrectly labeled examples in the dataset can also be viewed as number of successes in a sequence of m independent binary experiments each with a success probability λ. However, the n examples that are mislabeled in the dataset obtained from $IDCN(m, \eta_x, f, \mathcal{X}, \mathcal{D})$ are determined by function η_x whereas in the case of a dataset obtained from $NCN(m, \eta, f, \mathcal{X}, \mathcal{D})$ *any* n of the m instances can be mislabeled (For example the label of an instance x for which $\eta_x = 0$ will never be mislabeled in a dataset obtained from $IDCN(m, \eta_x, f, \mathcal{X}, \mathcal{D})$ whereas it is possible that the same instance can be mislabeled in a dataset obtained from $NCN(m, \eta, f, \mathcal{X}, \mathcal{D})$). The preceding argument leads to the following observation:

Observation 4: The IDCN model can be simulated by the NCN model and hence also by the NSN model.

The IDCN oracle uses a noisy oracle $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$ while the $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$ oracle uses a k-delegating oracle $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$. However, Theorem 1 states that every k-delegating Oracle has an *equivalent* Noisy Oracle. This leads to the following observation:

Observation 5: The $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$ Oracle can be simulated by the IDCN model.

Observation 4 and observation 5 results in a hierarchy of Oracles and is depicted in Figure 5.4.

It follows in a straightforward way from *Observation 4* and *Observation 5* (depicted in Figure 5.4) that learning in presence of ontology mapping errors can be seen as a weaker case of learning with nasty classification noise. As a result a learner can apply the same techniques to deal with mapping errors that it applies to deal with nasty classification noise. This result proves to be very useful in practice, since techniques to deal with noise have been studied extensively in literature and can be ported in a straightforward way to the setting of learning in presence of mapping errors. For example, similar to learning from noisy data, learning in presence of mapping errors is prone to overfitting and may be addressed by pruning [John (1995), Mansour (1997), Quinlan (1993)]. Similarly, on the lines of eliminating class noise (to improve performance of the learned classifier) filtering instances with mapping errors may be used to improve the performance of the classifiers learned in presence of mapping

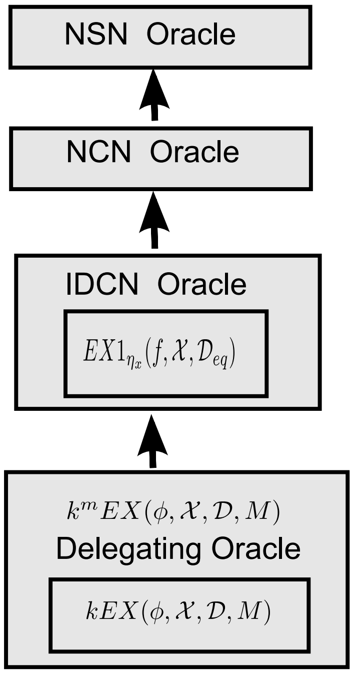
NSN Oracle
NCN Oracle
IDCN Oracle
$EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$
$k^m EX(\phi, \mathcal{X}, \mathcal{D}, M)$
Delegating Oracle
$kEX(\phi, \mathcal{X}, \mathcal{D}, M)$

Figure 5.4 A schematic representation of hierarchy between types of oracles (the arrows denote can be simulated by).

errors. Based on the approach in [Verbaeten and Assche (2003), Zhu et al. (2003), Gamberger et al. (1999)] , the training set D is partitioned into n subsets and a classifier built on each subdataset. A set of classifiers H_y from the aggregation of any $n-1$ subsets is used to classify the excluded subset, and the instances that are incorrectly labeled are marked as one with mapping errors and filtered out. This approach can be readily extended to detect mapping errors in the distributed setting using a technique similar to described by Zhu et al. (2006b). Other approach to filter class noise (and correspondingly mapping errors) include a boosting based filter [Verbaeten and Assche (2003), Karmaker and Kwek (2005)]. In this approach, after a certain number of iterations in AdaBoost [Freund and Schapire (1997)], instances whose weights exceed a a certain threshold are marked as having mapping errors. In addition, insights from noisy learning are also applicable to the setting of learning in presence of mapping errors, e.g. AdaBoost [Freund and Schapire (1997)], whose performance is known to degrade in presence of classification errors (see Dietterich (2000b)), is not a good choice to learn in the setting of mapping errors (a good choice may be Robust Alternating AdaBoost [Allende-Cid et al. (2007)], an noise-tolerant version of AdaBoost and hence also tolerant to mapping errors).

5.3.4 Learning in the Presence of Mapping Errors

We now proceed to present some theoretical results for learning in presence of mapping errors in a PAC like setting.

Definition 13 (PAC Learnability (from Kearns and Vazirani (1994))) *A class F of boolean functions is PAC-learnable using hypothesis class $\mathcal{H}$ in polynomial time if there exists an algorithm that, for any $f \in \mathcal{H}$, any $0 < \epsilon < 1/2$, $0 < \delta < 1$ and any distribution $\mathcal{D}$ on $\mathcal{X}$, when given access to the PAC oracle, runs in time polynomial in $log|\mathcal{X}|, 1/\epsilon, 1/\delta$ and with probability at least $1 - \delta$ outputs a function $h \in \mathcal{H}$ for which $Pr_{x \in \mathcal{D}}[h(x) \neq f(x)] \leq \epsilon$.*

Definition 14 (Mapping Error Rate) *The mapping error rate of a* k*-delegating oracle $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ is defined as the probability that an example $\langle x, \phi(x) \rangle$ obtained by making a call to $kEX(\phi, \mathcal{X}, \mathcal{D}, M)$ has a label that is different from that assigned by the target function f.*

Observation 6: The mapping error rate of a k-delegating oracle $kEX(\phi, \mathcal{X}, \mathcal{D}, M) = \sum_{x \in \mathcal{X}} \beta(x) \times Pr_{x \in \mathcal{D}}[x]$.

PAC learning is information theoretically impossible in the case when the probability that a randomly drawn example from an oracle has an incorrect label ≥ 0.5. Hence PAC learning is not possible in the case of Noisy oracle $EX1_{\eta_x}(f, \mathcal{X}, \mathcal{D}_{eq})$ when $\forall x, \eta_x > 0.5$ or in the case of $NCN(m, \eta, f, \mathcal{X}, \mathcal{D})$ when $\eta \geq 0.5$. Correspondingly, PAC learning is also not possible when the mapping error rate $\beta \geq 0.5$. This result provides an upper bound on the amount of mapping errors that can be tolerated. Consider the case of a k-delegating oracle in which mapping associated with one of the subordinate oracle has errors (we need not which them is erroneous). If in each run of the delegating oracle there is an equal chance of selecting each sub ordinate oracle, then $\beta \leq \frac{1}{k}$. Hence, for PAC learning, it needs to be the case that $k > 2$ which ensures $\beta < 0.5$.

Consider NastyConsistent, a PAC learning algorithm under the NSN model [Bshouty et al. (1999)].

Algorithm NastyConsistent

Input: certainty parameter $\delta > 0$, the nasty error rate $\eta < \frac{1}{2}$ and required accuracy $\epsilon = 2\eta + \Delta$.

begin

1. Request a sample $S = \{\langle x, l(x) \rangle\}$ of size $m > \frac{c}{\Delta^2}(d + log2/\delta)$ from the NSN oracle.

2. Output any $h \in \mathcal{F}$ such that $|\{x \in S : h(x) \neq l(x)\}| \leq m(\eta + \Delta/4)$ (if no such h exists, output any $h \in \mathcal{F}$).

end

Theorem 3 *(restatement of theorem 4 in Bshouty et al. (1999)) Let $\mathcal{C}$ be any class of VC-dimension d. Then, there exists a choice of the constant c for which NastyConsistent is a PAC learning algorithm under nasty sample noise of rate η.*

Proof 3 *See proof of Theorem 4 in Bshouty et al. (1999).*

Consider the following variant of the NastyConsistent algorithm which uses $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$ Oracle to return the sample S to the learner (as opposed to NCN Oracle in NastyConsistent).

Algorithm MappingErrorTolerantConsistent

Input: certainty parameter $\delta > 0$, the mapping error rate $\beta < \frac{1}{2}$ and required accuracy $\epsilon = 2\eta + \Delta$.

begin

1. Request a labeled dataset $S = \{\langle x, \phi(x)\}$ of size $m > \frac{c}{\Delta^2}(d + log2/\delta)$ from $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$.

2. Output any $h \in \mathcal{F}$ such that $|\{x \in S : h(x) \neq \phi(x)\}| \leq m(\eta + \Delta/4)$ (if no such h exists, output any $h \in \mathcal{F}$).

end

Theorem 4 *Let $\mathcal{C}$ be any class of VC-dimension d. Then, there exists a choice of the constant c for which* MappingErrorTolerantConsistent *is a PAC learning algorithm under the mapping error rate β.*

Proof 4 *The algorithm* MappingErrorTolerantConsistent *differs* NastyConsistent *in that it uses a $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$ instead of a NCN Oracle to get the labeled dataset. Since we have shown that the $kEX^m(\phi, \mathcal{X}, \mathcal{D}, M)$ can be simulated by the NCN oracle, the statement of the theorem follows from Theorem 2.*

We now proceed to present an open problem in the setting of learning in presence of mapping errors.

Definition 15 ***(Non-Trivial Concept Class (adapted from Bshouty et al. (1999)))*** *A concept class $\mathcal{F}$ over an instance space $\mathcal{X}$ is called non-trivial if there exist two points $x_1, x_2 \in \mathcal{X}$ and two concepts $f_1, f_2 \in \mathcal{F}$, such that $f_1(x_1) = f_2(x_1)$ and $f_1(x_2) \neq f_2(x_2)$.*

Consider the following negative result concerning PAC learnability in the NCN model. **Theorem**(restatement of theorem 3 in Bshouty et al. (1999)) Let $\mathcal{C}$ be a non-trivial concept class, η be a noise rate and $\epsilon \leq 2\eta$ be an accuracy parameter. Then, there is no algorithm that learns the concept class $\mathcal{C}$ with accuracy ϵ under the NCN model (with rate η).

This raises the open question as to whether such a non-trivial concept class is PAC learnable in the restricted case of IDCN and hence in the presence of mapping errors.

5.4 Summary and Discussion

5.4.1 Significance

The rapid proliferation of autonomous, distributed data sources in many emerging data-rich domains (e.g., bioinformatics, social informatics, security informatics) coupled with the rise in the use of ontologies to associate semantics with the data has led to a growing interest in the problem of learning predictive models from *semantically disparate* data sources. Many practical approaches to this problem rely on *mapping* the instance descriptions used by the individual data sources into instance descriptions expressed in a common representation assumed by the learner (As an example G02 (2009) lists mappings between 20 different ontologies to the gene ontology). Establishing such mappings is a complex and inevitably error-prone process. Hence there is a need for approaches to learning from such data in the presence of mapping errors.

In this paper we have established that the problem of learning from semantically disparate data sources in the presence of mapping errors can be reduced to the problem of learning from a single data source in the presence of nasty classification noise within a PAC-like framework. It should be noted that reduction to any arbitrary noise model is not applicable. For example, in general, learning in the presence of mapping errors **cannot** be reduced to the problem of learning in presence of random classification noise. In the random classification noise model, the label of each instance can get flipped with a fixed probability η. In contrast, in the case of a k-delegating oracle model, a given instance always gets assigned the same label. This is because the mappings regardless of whether they are correct or not are fixed prior to sampling and will result in an instance (when sampled) always being assigned the same label. Hence, it is possible that a dataset D generated from an Oracle with random classification noise can include two examples of the form $\langle x, 0\rangle$ and $\langle x, 1\rangle$ (i.e. D contains the same instance with two different labels). The dataset D can never be generated by a k-delegating oracle model since it will always label the instance x in the same way (even if x occurs multiple times in D).

The reduction of learning in the presence of mapping errors and learning in the presence of nasty classification noise opens up the possibility of applying existing results and approaches to learning in presence of classification noise to the problem of learning in the presence of mapping errors. In our opinion this reduction is important since it provides a theoretical basis for solving practical issues that arise in learning in the semantically disparate setting. Based on this reduction, we outlined some of the techniques that can be used to cope with errors in mappings in this setting. We believe these techniques will prove do to be very useful in practice as the use of ontologies becomes even more widespread. On a theoretical side, we also presented an algorithm that can be used to learn in presence of mapping errors in a PAC like setting.

5.4.2 Related Work

There is growing interest in the problem of learning predictive models from distributed data sources [Park and Kargupta (2003)]. Caragea et al. (2005) have described algorithms that given correct mappings, provide rigorous performance guarantees (relative to their single data source counterparts) for learning from distributed, semantically disparate data sources when the mappings are semantic preserving. Crammer et al. (2008) have examined the problem of learning predictors from a set of *related* data sources. Ben-david et al. (2002) have analyzed the sample complexity of learning from semantically disparate data sources in a setting where classifiers trained on data sources $D_1 \cdots D_{n-1}$ are used to predict the class labels of instances from a data source D_n. However, none of these works have considered the effect of errors in mappings between the representations used by the individual data sources.

The problem of learning predictive models from in the presence of noise in the data has received considerable attention in the literature. A variant of PAC learning to model learning in the presence of random classification noise was introduced in [Angluin and Laird (1998)]. Other variants of PAC learning that have been considered to model learning from noisy data include PAC learning in the presence of malicious errors [Kearns and Li (1993)], learning in the presence of attribute noise (but not classification noise) [Shackelford and Volper (1988), Goldman and Sloan (1995)], learning under the nasty (or adversarial) noise model [Bshouty et al. (1999)]. Several different types of noise in data have been been examined in the context of the PAC learning framework in Sloan (1995). A quantitative study of classification noise and attribute noise is given in Zhu and Wu (2004). Wilson and Martinez (2004) provide an overview of approaches to cope with noise in data. Karmaker and Kwek (2005) have described a boosting based approach to detect outliers in data which is closely related to the problem of detecting mislabeled examples in a noisy dataset.

There has been very little work on detecting mapping errors in the setting of learning from heterogeneous data sources. Of related interest is the work in ontology mapping field [Kalfoglou and Schorlemmer (2005), Euzenat and Shvaiko (2007)]. However, the primary focus in this area is aligning ontologies (through use of mappings), merging related ontologies or detecting logical inconsistencies in mappings [Meilicke et al. (2007)]. However, a consistent mapping need not be correct in the sense described in this paper and in addition the focus of this paper is to learn in presence of mapping errors.

5.4.3 Future Work

There are several interesting directions along which the analysis presented in this paper can be extended including in particular, consideration of the effect of mapping errors in multi-relational learning multiple instance learning, multi-label and structured label learning, among others. Also of interest are theoretical and experimental studies of alternative approaches to learning from semantically disparate data in the presence of mapping errors.

CHAPTER 6. RELATIONSHIP BETWEEN LEARNING CLASSIFIERS FROM DISPARATE DATA AND DOMAIN ADAPTATION

Current approaches to learning predictive models (e.g., classifiers) from semantically disparate data sources rely on mappings between the corresponding data source ontologies (i.e., logical specification of abstractions of the underlying data model that capture knowledge of individuals, their attributes, and their relationships to other individuals). Such mappings should ideally establish semantic correspondences between the learner's data model and the data models associated with the individual sources. Because specifying such mappings, or for that matter, even choosing an optimal mapping from among as set of alternative mappings is a tedious and often error-prone process, and because the accuracy of the learned predictive model depends on the quality of the mappings used, it is important to understand (a) how errors in mappings affect the accuracy of the learned model and (b) how to choose an optimal mapping from among a set of alternative expert-supplied or automatically generated mappings. Towards this end, we introduce a notion of reducibility among classes of supervised learning tasks and show that several aspects of learning from semantically disparate data sources can be reduced to, and hence understood in terms of the theoretically well-studied problem of domain adaptation (i.e., adapting a model that is trained on data sampled according to a distribution that is different from the distribution from which the test data are obtained). Furthermore, we introduce the notion of probabilistic mappings and show that there exists a specific probabilistic mapping that facilitates domain adaptation.

6.1 Introduction

The proliferation to autonomous, and hence often semantically disparate data sources presents several significant challenges in learning from data [Caragea et al. (2005); Honavar and Caragea (2008)]. As a result,different data sources often use disparate vocabularies (e.g., M.S. student versus Masters student), measurement units (e.g., temperature measured in degrees Centigrade versus Fahrenheit), and levels of abstraction (e.g. graduate student, student) to describe the individuals of interest in the world being modeled. Hence, learning a predictive model from such multiple data sources based on disparate (yet semantically related) data models requires reconciliation of their semantic differences between the corresponding ontologies (i.e., logical specifications of abstractions of the respective data models that capture knowledge of individuals, their attributes, and their relationships to other individuals) and the ontology that corresponds to the learner's conceptualization of the world. This reconciliation is often achieved through the use of *mappings* that establish semantic correspondences between the learner's data model and the data models associated with the individual sources [Caragea et al. (2005); Kalfoglou and Schorlemmer (2005); Doan et al. (2002)]. Such mappings are either provided by a domain expert with an intimate knowledge of both the learner's ontology as well the data source ontologies or generated using automated techniques for ontology alignment [Euzenat and Shvaiko (2007)]. As we shall show later, such mappings influence the the attributes as well as distribution of samples seen by the learning algorithm, and thus, ultimately, the predictive model that is learned. Because the task of establishing mappings between disparate ontologies is necessarily an error-prone process, it is important to understand how errors in a mapping M_U relative to a "correct" or "ideal" mapping M_{true} impact the accuracy of the learned model. A related problem has to do with choosing an optimal mapping from among a set of available candidate mappings (either provided by different experts or generated using one of the several available ontology alignment methods). To answer these questions, we introduce a notion of reducibility among classes of supervised learning problems and use it to show that several aspects of learning from semantically disparate data sources can be reduced to, and hence understood in terms of the theoretically well-studied problem of domain adaptation (i.e., adapting a model that is trained on data sampled according to a distribution that is different from the distribution from which the test data are obtained) [Blitzer et al. (2007a); Mansour et al. (2008); Daumé III and Marcu (2006)]. Furthermore, we show that there exists a transformation (which we call a probabilistic mapping) between the instances spaces of two different domains that facilitates domain adaptation. These results allow the transfer of theoretical results as well as practical algorithms between two areas of data mining that have so far been studied separately: learning predictive models from semantically disparate data sources and domain adaptation.

The rest of the chapter is organized as follows. Section 6.2 describes an approach to model various classes of supervised learning tasks. Section 6.3 introduces the notion of reducibility among classes of supervised learning tasks and uses it to establish relationships between aspects of learning predictive models from semantically disparate data and aspects of domain adaptation. Section 6.4 shows how probabilistic mappings can be used for domain adaptation. Section 6.5 concludes with summary, discussion of related work, and a brief outline of some directions for further research.

6.2 Modeling Learning Tasks

We first introduce some notation that we will use throughout rest of the chapter. Let $\mathcal{X}$ be an instance space and $\mathcal{H}$ be an hypothesis space. In *supervised learning* the task of a learner $\mathcal{L}$ is to learn a target function $f : \mathcal{X} \mapsto \{0,1\}$ given a set of instances drawn from $\mathcal{X}$ (according to a fixed but unknown distribution $\mathcal{D}$) and labeled according to f. Note for simplicity we restrict our setting to binary classification with class labels 0 and 1. Given a training dataset $S \subseteq \mathcal{X} \times \{0,1\}$ the learner $\mathcal{L}$ outputs a hypothesis $h \in \mathcal{H}$ so as to minimize some error function. Let $S_{\mathcal{X}}$ denote the subsets of $\mathcal{X} \times \{0,1\}$. Hence, a learning algorithm $\mathcal{L}$ can be seen as function $\mathcal{L} : S_{\mathcal{X}} \mapsto \mathcal{H}$ that maps a training dataset to a hypothesis in $\mathcal{H}$. Given a training dataset $S \subseteq \mathcal{X} \times \{0,1\}$ we also denote the hypothesis output by the learner using the training dataset S as $\mathcal{L}(S)$. In our setting, the error of a hypothesis h (output by $\mathcal{L}$) is denoted by $\epsilon_{\mathcal{D}_t}(h, f)$ and is the expectation that hypothesis h and f do not label an example drawn randomly from $\mathcal{X}$ according to distribution $\mathcal{D}_t$. We use the notation $Pr_{x \in \mathcal{D}_t}[x]$ to indicate the probability of drawing an instance x from $\mathcal{X}$ according to the test distribution $\mathcal{D}_t$ (which, in general, may differ from the training distribution $\mathcal{D}_s$). Similarly we use $E_{x \in \mathcal{D}_t}(x)$ to indicate the expectation of x taken with respect to probability measure $\mathcal{D}_t$. It follows that $\epsilon_{\mathcal{D}_t}(h, f) = E_{x \in \mathcal{D}_t}(|h(x) - f(x)|)$.

Let $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ denote an oracle that when invoked by a learner $\mathcal{L}$ returns a labeled example $\langle x, f(x)\rangle$ where x is drawn from $\mathcal{X}$ according to a fixed but unknown distribution$\mathcal{D}$. Supervised learning is traditionally modeled by assuming that the learner $\mathcal{L}$ has access to the oracle $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ that the learner $\mathcal{L}$ invokes multiple time to obtain a training dataset [Kearns and Vazirani (1994)]. The said approach of a learner $\mathcal{L}$ with access to the oracle $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ can model neither domain adaptation nor learning from semantically disparate date sources. As such, in the next section, we present an extension to the classical model of supervised learning that allows to model both learning from semantically disparate data sources as well as domain adaptation. Subsequentally, we use the said model to show that several aspects of learning from semantically disparate sources can e reduced to the well studied problem of domain adaptation.

6.2.1 Learning from Semantically Disparate Data Sources

Let us introduce a very simple example to bring forth the setting of learning from semantically disparate data sources. Consider the problem of *learning boolean conjunctions.* The target function $f : \{0, 1\}^n \longrightarrow \{0, 1\}$ to be learned is a conjunction of literals drawn from $x_1, x_2 \ldots x_n$ and their negations. The learner $\mathcal{L}$ expects the training examples of the form (x, y) where $x \in \{0, 1\}^n$ and $y \in \{0, 1\}$. Suppose the learner receives the training data (labeled examples) from two different sources D_1 and D_2. Let each instances from D_1 (as well as D_2) correspond to an n valued attribute value vector, and an ontology associated with each attribute (and the label) specify the possible values that the attribute can take. Let the ontology associated with each attribute and the class label in D_1 be such that each attribute as well as the class label can take either then boolean value *True* or *False.* Hence, the data from

D_1 is of the form $(x,\ y)$ where $x \in \{True, False\}^n$ and $y \in \{True, False\}$. Similarly, let the ontologies associated with the attributes and the class label in D_2 be such that the data from D_2 is of the form $(x,\ y)$ where $x \in \{-5V, +5V\}^n$ and $y \in \{Off, On\}$ (Note data in D_2 can be seen as record of the input voltages, and output to boolean circuit that encodes the function f). Now in order for the learner to be able to learn from data from both D_1 and D_2, the vocabularies used by D_1 and D_2 have to be mapped to the vocabulary used by the learner. Thus, suppose -5V, +5V, *Off*, *On* in D_2 map to 0, 1, 0, 1 (respectively) from the learner's point of view. Similarly, $False$ and $True$ in D_1 *map to 0 and 1*(respectively) from the learner's point of view. The described mappings between ontologies can be used to construct function that transform examples from D_1 and D_2 into examples from which $\mathcal{L}$ can learn the unknown target function *f.* While in this example, the mappings were relatively simple, in real-world applications, establishing such mappings and ensuring that they preserve the *intended* semantics can be a complex, and error-prone process. Suppose in our example because of human error, the *On* is incorrectly mapped to *0* instead of 1. As a consequence of this mapping error, some of the instances from D_1 are incorrectly labeled from the learner's point of view. Hence, in general, the learner $\mathcal{L}$ must be robust in presence of mapping errors. Another problem the occurs in this setting is the problem of multiple available mappings. Consider two mappings (say developed independently by two competing groups) where in the first -5V, +5V, *Off*, *On* in D_2 is mapped to 0, 1, 1, 0 (respectively) in the learner's view whereas the second mapping (say obtained from another group) maps *Off*, *On* in D_2 to 1, 0, 0, 1 (respectively) in learner's view. As such the learner need to handle this scenario of multiple available mappings (say by choosing the best among the multiple available mappings).

It is straightforward to observe that the *classical* oracle $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ cannot model examples being drawn in the setting of learning from disparate data sources. We introduce a *kDelegating oracle* , an extension of the classical oracle $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$, that can be used to model learning from disparate data sources.

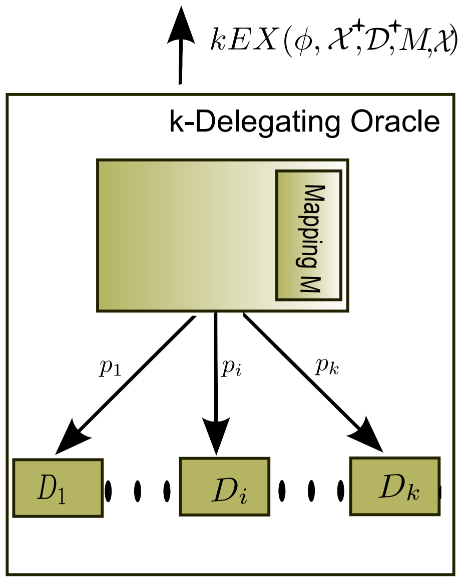

Figure 6.1 A *kDelegating* oracle with D_i denoting the oracle $EX^i\langle f_i, \mathcal{X}_{s^i}, \mathcal{D}_i\rangle$.

6.2.2 *k*Delegating oracle

A *kDelegating oracle* is a procedure which when invoked by a learner in turn invokes subordinate oracles $EX^1\langle f_1, \mathcal{X}_{s^1}, \mathcal{D}_1\rangle \ldots EX^k\langle f_k, \mathcal{X}_{s^k}, \mathcal{D}_k\rangle$ with probabilities $p_1 \ldots p_k$ respectively. The *kDelegating oracle* has access to a mapping set $M = \{m_1, m_2 \ldots m_k\}$ where $m_i = \{m_i^x, m_i^c\}$; $m_i^x : \mathcal{X}_{s^i} \longrightarrow \mathcal{X}$ is an *attribute mapping function*; and $m_i^c : C_i \longrightarrow C$ is a *class mapping function* where $C_i = Range(f_i)$ and $C = Range(f)$. It uses the mapping m_i to convert an instance $\langle x_{s^i}, f_i(x_{s^i})\rangle$ received form the i^{th} subordinate oracle to $\langle m_i^x(x_{s^i}), m_i^c(f_i(x_{s^i}))\rangle$ before passing it to the learner. A mapping aware k-delegating oracle is represented by $EX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$, where $\mathcal{X}^+ = \{\mathcal{X}_1, \mathcal{X}_2 \ldots \mathcal{X}_k\}$ is the set of instance spaces, $\mathcal{D}^+ = \{\mathcal{D}_1, \mathcal{D}_2 \ldots \mathcal{D}_k\}$ is the set of distributions, M is a mapping set and ϕ is the labeling function with which instances are returned to the learner. A schematic representation of learning using a *kDelegating* oracle is shown in Figure 6.1.

The mapping $m_i = \{m_i^x, m_i^c\}$ is said to be admissible if it satisfies the following conditions:

$\forall x_{s^i} \in \mathcal{X}_{s^i}, m_i^x(x_{s^i}) \in \mathcal{X}$; $\forall l \in Range(f_i)$, $m_i^c(l) \in Range(f)$ and whenever $x \in \mathcal{X}_{s^i}, \mathcal{X}_{s^j}$, $m_i^x(x) = m_j^x(x)$. A mapping set M is said to be admissible if $\forall m_i \in M$, m_i is admissible. An admissible mapping set M ensures that the examples returned by the k-delegating oracle are of the form $\langle x, l(x)\rangle$ where $x \in \mathcal{X}$ and $l(x) \in C$. We denote the set of all possible admissible mappings by $\mathcal{M}*$ and assume in the rest of the chapter that a given mapping is admissible. Consider the case of learning a function f using a kDelegating oracle $kEX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$. A straight-forward analysis shows that the total number of admissible mappings is $|\mathcal{M}*| = \prod_i^k |\mathcal{X}|^{|\mathcal{X}_{s^i}|}$ (we assume two mappings are different if they differ on mapping atleast one instance from the instance space of any one of the subordinate oracles).

Let the ideal mapping set $M_{true} = \{m_{1,true}, m_{2,true} \ldots M_{k,true,}\}$ where $m_{i,true} = \{m_{i,true}^x, m_{i,true}^c\}$. The ideal mapping set is assumed to satisfy the following conditions: (a) $M_{true} \in \mathcal{M}*$ and (b) $\langle x, l(x)\rangle$ the labeled instance returned to $\mathcal{L}$ (obtained after applying relevant mappings in M_{true} to the labeled instance sampled from the subordinate oracle) is the same as $\langle x, f(x)\rangle$. The condition (b) requires that instances returned to the learner are labeled according to the target function f.

Definition 2 ***Semantics Preserving Class Mapping :*** *A class mapping m_i^c is said to be semantics preserving if $\forall l \in C_i$ $m_i^c(l) = m_{i,true}^c(l)$.*

Definition 3 ***Semantics Preserving Attribute Mapping:*** *An attribute mapping m_i^x is said to be semantics preserving whenever $\forall x \in \mathcal{X}_{s^i}$, $f_i(x) = l$ and $m_{i,true}^c(l) = l_1$ implies $f(m_i^x(x)) = l_1$*

Definition 4 ***Semantics Preserving Mapping Set :*** *A mapping set $M = \{m_1, m_2 \ldots m_k\}$ is said to be correct if $\forall i \in \{1, 2, \ldots, k\}$ m_i^x and m_i^c are semantics preserving.*

Definition 5 ***Identity Mapping*** *A mapping set M is said to be an identity mapping if $\forall m_i = \{m_i^x, m_i^c\} \in M, m_i^x(x) = x$ and $m_i^c(l) = l$.*

A semantics preserving mapping guarantees that the instances returned to the learner are labeled with the target function f. Hence, given a kDelegating oracle $kEX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$

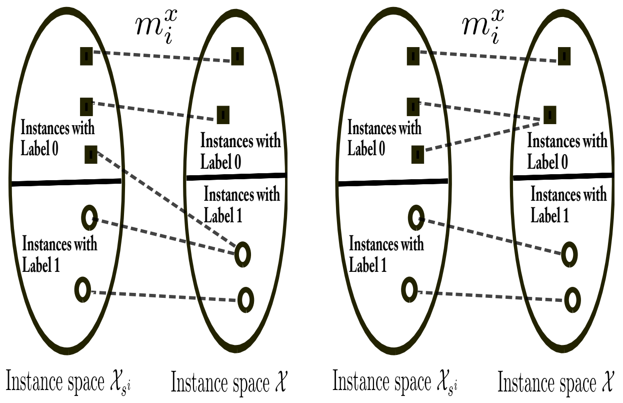

Figure 6.2 An example of a mapping with errors and a semantics preserving mapping.

with semantics preserving mapping M, it is the case that $\phi = f$ where f is the target function. It is the case that M_{true} is semantics preserving. A mapping set M is said to have errors if it is not semantics preserving. Errors in a mapping set can either be due to errors in attribute mappings or errors in class mappings. Since the class mappings are fairly straightforward (being between the binary class labels 0 and 1), we will assume throughout the chapter that only possible errors are in attribute mappings. Figure 6.2 shows an example mapping with errors and a semantics preserving mapping.

Remark A straight-forward analysis shows that the total number of semantics preserving mappings is $\prod_i^k |\mathcal{X}^+|^{|\mathcal{X}_{si}{}^+|} \times |\mathcal{X}^-|^{|\mathcal{X}_{si}{}^-|}$.

The remark above shows that, in general, there are multiple available semantics preserving mappings each of which returns examples to the learner that are labeled with the target function. We now show that difference between using different semantics preserving mappings manifests itself in terms of difference in distribution with which examples are returned to the learner. Given a mapping m_i^x, let $[m_i^x(y \mapsto z)]$ be a indicator function that returns 1 when $m_i^x(y) = z$ (for $y \in \mathcal{X}_{si}$) and returns 0 otherwise. Given a kDelegating oracle $kEX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X} \rangle$ the distribution over $\mathcal{X}$ with which instances are returned to the learner $\mathcal{L}$ is

$$Pr[z] = \sum_{i=1}^{k} p_i \sum_{y \in \mathcal{X}_{si}} [m_i^x(y \mapsto z)] pr_{y \in \mathcal{D}_i}[y]$$

Hence, given a kDelegating oracle $kEX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X} \rangle$ the distribution with which instances are returned to the learner $\mathcal{L}$ depends on the set $\mathcal{P} = \{p_1 \dots p_k\}$ (the probabilities of selecting the subordinate oracles), the distribution set $\mathcal{D}^+$ and the mapping set M (through dependence on indicator function $[m_i^x(y \mapsto z])$ and is denoted by $M_{\mathcal{P}}(\mathcal{D}^+)$. For easing clutter of notation we drop the parameter $\mathcal{P}$ in the notation and denote it by $M(\mathcal{D}^+)$.

While a learner $\mathcal{L}$ with access to a *kDelegating* oracle can easily model learning from semantically disparate data (including modeling errors in mappings), it is the case that a learner with access to a *kDelegating* oracle cannot model domain adaptation. We now introduce, in the next section, an approach to model domain adaptation.

6.2.3 Modeling Domain Adaptation

Consider a learner $\mathcal{L}$ whose task is to a learn a target function $f : \mathcal{X} \mapsto \{0, 1\}$ using an oracle $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$. In such a case $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ denotes a family of oracles for the various values of the parameter $\mathcal{D}$. A specific value of $\mathcal{D}$ defines a specific instantiated oracle. Similarly, for a kDelegating oracle $kEX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$ various values of the parameters $\mathcal{X}^+, \mathcal{D}^+$ and M describe specific instantiated oracle. We use the notation $EX\langle\theta\rangle$ to denote a family of oracles with the parameter θ. The notation, $EX\langle\theta\rangle$, will be used to denote both the classical oracle and the kDelegating oracle. We use the notion of family of oracles to describe a class of supervised learning problems.

Definition 6 ***Learning Scenario***: *A learning scenario for a learner $\mathcal{L}$ is a two-tuple $\mathcal{T} = \{EX\langle\theta_1\rangle, EX\langle\theta_2\rangle\}$ where the during the training phase the learner $\mathcal{L}$ has access to labeled examples drawn from the oracle $EX\langle\theta_1\rangle$ (called the training oracle) while the error of the hypothesis output by $\mathcal{L}$ at the end of the training phase is computed with respect to an example to be drawn from oracle $EX\langle\theta_2\rangle$ (called the test oracle). The error of the hypothesis computed by $\mathcal{L}$ in this setting is denoted by $R(\theta_1, \theta_2)$.*

Intuitively, the *Learning Scenario* can be seen as traditional classroom teaching , wherein the teacher teaches the course material to the students and then tests their performance by means of an examination.

A learning scenario, in the case, where the training and test oracles represent a family oracles corresponds to a class of problems. An instantiation of the training and the test oracles defines a specific problem in this class. The notion of the learning scenario is powerful enough to model a significant class of supervised learning tasks. We describe below the learning scenarios that correspond to some well known types of supervised learning.

- **Classical Supervised learning**: This corresponds to $\mathcal{T} = (EX\langle f, \mathcal{X}, \mathcal{D}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}\rangle)$.

- **Distributed Learning**: This corresponds $\mathcal{T} = (kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, EX(f, \mathcal{X}, \mathcal{D}))$ where for the kDelegating training oracle the i^{th} subordinate oracle is $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ and M is an identity mapping.

- **Learning from disparate data sources** [Caragea (2004); Ben-david et al. (2002)]: This corresponds to the learning scenario $\mathcal{T} = (kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}\rangle)$ where for the training oracle the i^{th} subordinate oracle is $EX\langle f, \mathcal{X}_{s^i}, \mathcal{D}_i\rangle$.

- **Learning Under Covariate Shift and Sample Selection Bias**: Learning under Covariate shift [Bickel et al. (2009)] corresponds to $\mathcal{T} = (EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$. Sample selection bias [Zadrozny (2004); Heckman (1979); Cortes et al. (2008)] which corresponds to the setting where each training instance is originally drawn from the test distribution, but is then selected into the training sample with some probability, or discarded otherwise can also be modeled by the same learning scenario.

- **Learning With Auxiliary Data Sources** [Wu and Dietterich (2004); Liao et al. (2005)]: This corresponds to learning scenario $\mathcal{T} = (2EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$ where M is an identity function. The two subordinate oracles of the 2Delegating test oracle are $EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle$ and $EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle$ with $\mathcal{P} = \{p, 1-p\}$ controlling the amount of auxiliary data available. Recall the elements of the set $\mathcal{P}$ represents the probabilities of selecting the subordinate oracles.

- **Domain Adaptation** [Ben-David et al. (2007); Blitzer et al. (2007b)]: This corresponds to the learning scenario $\mathcal{T} = (EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle)$.

- **Transfer Learning**: This corresponds to the learning scenario $\mathcal{T} = EX\langle f_s, \mathcal{X}_s, \mathcal{D}_s\rangle$ and $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$ (see survey Pan and Yang (2010)).

The learning scenario can be easily extended to the setting of *Semi-Supervised Learning* [Chawla and Karakoulas (2005); Chapelle et al. (2006); Singh et al. (2008)] where both labeled and unlabeled data is used in learning. This is achieved by introducing the notion of specialized subordinate oracle in the kDelegating oracle that returns unlabeled examples. It is interesting to note the use of a learning scenario allows to model classes of problems that have not been explicitly mentioned in literature. For example, $\mathcal{T} = (2EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$ where M is an semantics preserving admissible mapping corresponds to the problem of *adapting semantically disparate domains.* We now proceed to introduce the notion of reducibility between learning scenarios.

6.3 Reducibility between learning tasks

Definition 7 ***Indistinguishability*** *For a learner $\mathcal{L}$ whose task is to learn a function $f : \mathcal{X} \mapsto \{0, 1\}$ two instantiated oracles $EX\langle\theta_1\rangle$ and $EX\langle\theta_2\rangle$ are said to be indistinguishable if $\forall S \in \mathcal{X} \times \{0, 1\}$ the probability of the dataset S being drawn with $|S|$ calls to oracle $EX\langle\theta_1\rangle$ or oracle $EX\langle\theta_2\rangle$ is the same.*

Intuitively, given two indistinguishable oracles $EX\langle\theta_1\rangle$ and $EX\langle\theta_2\rangle$, a learner $\mathcal{L}$ will not be able to distinguish whether a given dataset S was drawn from $EX\langle\theta_1\rangle$ or $EX\langle\theta_2\rangle$. This notion is useful since a techniques that a learner applies to learn using an oracle $EX\langle\theta_1\rangle$ (and any applicable bounds and guarantees) will port in a straightforward manner to learning using the oracle $EX\langle\theta_2\rangle$.

Definition 8 ***Subclass Oracles*** *For a learner $\mathcal{L}$, a family of oracles $EX\langle\theta_1\rangle$ is said to be subclass of family of oracles $EX\langle\theta_2\rangle$, denoted by $EX\langle\theta_1\rangle \subset EX\langle\theta_2\rangle$, if for all possible instantiations of the parameters θ_1 there exists an instantiation of the parameters θ_2 such that the corresponding instantiated oracles $EX\langle\theta_1\rangle$ and $EX\langle\theta_2\rangle$ are indistinguishable with respect to the learner $\mathcal{L}$.*

The following observation follows directly from the definition of subclass Oracles: $EX\langle\theta_1\rangle \subset EX\langle\theta_1\rangle$.

Definition 9 ***Reducibility*** *A learning scenario $\mathcal{T}_1 = (EX\langle\theta_1\rangle, EX\langle\theta_2\rangle)$ is said to be reducible to a learning scenario $\mathcal{T}_2 = (EX\langle\theta_3\rangle, EX\langle\theta_4\rangle)$, denoted by $\mathcal{T}_1 \subset \mathcal{T}_2$, if $EX\langle\theta_1\rangle \subset EX\langle\theta_3\rangle$ and $EX\langle\theta_2\rangle \subset EX\langle\theta_4\rangle$*

Reducing a learning scenario $\mathcal{T}_1$ to a learning scenario $\mathcal{T}_2$ shows that, in general, that problem $\mathcal{T}_2$ is as hard as $\mathcal{T}_1$. As a result techniques to solve $\mathcal{T}_2$ can be used to solve $\mathcal{T}_1$. In addition any negative results applicable to $\mathcal{T}_1$ are also applicable to $\mathcal{T}_2$.

6.3.1 Learning when available mapping is different from the true mapping

We now proceed to show that learning with a semantics preserving mapping (say $M_U \in \mathcal{M}*$) instead of a true mapping M_{true} is reducible to the Domain Adaptation problem. We first present the following lemma.

Lemma 5 *Learning under Covariate Shift is reducible to Domain Adaptation. Formally, $\mathcal{T}_1 = (EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle) \subset \mathcal{T}_2 = (EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle)$*

Proof 5 *Since that task of the learner $\mathcal{L}$ is to learn a function $f : \mathcal{X} \mapsto \{0,1\}$, it follows that for the family of oracles $EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle$ the parameters are $\mathcal{D}_s$ and f_s. Note that f_s is a parameter since the function to be learned is f and not f_s. Hence $EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle \subset EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle$ corresponding to $f_s = f$. Similarly, $EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle \subset EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle$ corresponding to $f_t = f$. Hence, by definition of reducibility between learning scenarios, the statement of the lemma follows.*

Lemma 6 *Learning from disparate data sources using a semantics preserving mapping (M_U) that is different from the true mapping M_{true} is reducible to learning under covariate shift. Formally, $\mathcal{T}_3 = (kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle, kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle) \subset \mathcal{T}_2 = (EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$*

Proof 6 *Consider an instantiation of the oracle $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle$ where M_U has been chosen from the set of semantics preserving mappings. Consider a corresponding instantiation of the oracle $EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle$ where $\mathcal{D}_s = M_U(\mathcal{D}^+)$. Hence, $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle$ (with respect to the learner $\mathcal{L}$) since both draw instances from $\mathcal{X}$ according to distribution $M_U(\mathcal{D}^+)$ and label them according to f. Hence, $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle \subset EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle$. Similarly, $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle \subset EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle$. Hence, by definition of reducibility between learning scenarios, the statement of the lemma follows.*

Theorem 7 *Learning with a user provided semantics preserving mapping M_U (that is assumed to be different from the true mapping M_{true}) is reducible to domain adaptation. Formally, $\mathcal{T}_1 = (kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle, kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle) \subset \mathcal{T}_2 = (EX\langle f_s, \mathcal{X}, \mathcal{D}_s, \rangle, EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle)$*

Proof 7 *The statement of the proof follows directly from from lemma 5 and lemma 6 and the following observation: given learning scenarios $\mathcal{T}_1$, $\mathcal{T}_2$ and $\mathcal{T}_3$, it follows that $\mathcal{T}_3 \subset \mathcal{T}_2$ and $\mathcal{T}_2 \subset \mathcal{T}_1$ implies that $\mathcal{T}_3 \subset \mathcal{T}_1$.*

The reduction of the learning using an user provided semantics preserving mapping M_U that is different from true mapping M_{true} to domain adaptation turns out to be useful as it implies that any positive bounds applicable to domain adaptation (see Ben-David et al. (2010); Mansour et al. (2009); Blitzer et al. (2007a)) are applicable to this setting.

6.3.2 Learning in presence of Mapping errors

We now address the problem of learning when the available admissible mapping has errors by reducing it to the problem of learning in presence of nasty noise Bshouty et al. (1999).

Definition 10 ***(Nasty Sample Noise (NSN) oracle (adapted from Bshouty et al. (1999))):*** *A Nasty Sample Noise oracle, denoted by $NSN\langle m, \eta, f, \mathcal{X}, \mathcal{D}\rangle$, is one where an adversary obtains a dataset D^m of m i.i.d examples by making m calls to a classical oracle $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$. The adversary then picks n out of m instances of its choosing from D^m (where*

n is distributed according to a binomial distribution with parameters m and nasty noise rate η) and replaces them with any examples of its choice from $\mathcal{X} \times Range(f)$. The resulting dataset is then provided to the learner.

Definition 11 *(k^m-**delegating oracle**): A k^m-delegating oracle, denoted by $kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$, is one where an intermediary obtains a dataset D^m of m i.i.d examples by making m calls to a* k*-delegating oracle $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$. The resulting dataset is then provided to the learner.*

Theorem 8 *Learning in presence of mapping with errors is reducible to learning with nasty noise. Formally, The learning scenario $\mathcal{T}_1 = (kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle) \subset \mathcal{T}_2 = (NSN\langle m, \eta, f, \mathcal{X}, \mathcal{D}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}\rangle)$.*

Proof 8 *Let the instance space of the i^{th} subordinate oracle $\mathcal{X}_{s^i} = \{x_1, x_2 \ldots x_{|\mathcal{X}_{s^i}|}\}$. The mapping $M = \{m^x, m^c\}$ where $m^x = \{m^x_1, m^x_2 \ldots m^k_i\}$. Let $\Omega(m^x_i) = \{m^1_i, m^2_i \ldots m^{|\mathcal{X}_{s^i}|}_i\}$ be the set of atomic attribute mappings such that when the instance $\langle x_j, l(x_j)\rangle$ is sample from the i^{th} subordinate oracle, the applicable atomic mapping m^j_i is exercised to convert the instance x_j into the instance space $\mathcal{X}$ and is labeled by ϕ before returning to the learner. Let $\Omega(M) = \cup^k_{i=1}\Omega(m^x_i)$. Similarly for the true mapping M_{true} we have the set $\Omega(M_{true})$. Let $\Omega(M+)$ be the subset of atomic mappings in $\Omega(M)$ which when exercised to the applicable instance have the same result as when the corresponding atomic mapping is applied from the set $\Omega(M_{true})$. Then $\Omega(M-) = \Omega(M) - \Omega(M+)$ is the subset of atomic mappings in $\Omega(M)$ which differ in behavior from the corresponding mappings in $\Omega(M_{true})$. Intuitively, $\Omega(M+)$ is the part of the mapping M that is same as the true mapping M_{true}.*

Consider an instantiation of the oracle $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$. An atomic mapping in $\Omega(M)$ is exercised every time an example is returned to the learner by a single run of the oracle $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$. As a result a distribution is induced on the set $\Omega(M)$. Let D_M denote the resulting induced distribution on the set $\Omega(M)$. Let $\beta = \sum_{m\in\Omega(M-)} Pr_{m\in D_M}[m]$. Hence β can be seem as the probability that in a given run of the oracle $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$ the element returned to $\mathcal{L}$ is different than if M_{true} was used. Hence, in the dataset obtained *from $kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$ the number of times an atomic mapping in $\Omega(M)$ was exercised that was different from the corresponding atomic mapping in $\Omega(M_{true})$ can be seen as the number of success in a sequence of m independent binary experiments each with a probability β. Now consider an instantiation of $NSN\langle m, \eta, f, \mathcal{X}, \mathcal{D}\rangle$ where $\mathcal{D} = M_{true}(\mathcal{D}^+)$ and $\eta = \beta$. In this setting the number of samples that can be modified by the adversary can be seen as number of successes in a sequence of m independent binary experiments each with a probability η. Since $\eta = \beta$ and because the adversary can modify these examples in any possible way, the learner cannot distinguish whether the examples were modified by using an incorrect mapping or by the adversary (which can mimic the behavior of the atomic mappings in $\Omega(M-)$). Hence, $kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle \subset NSN\langle m, \eta, f, \mathcal{X}, \mathcal{D}\rangle$ i.e. the training oracle in $\mathcal{T}_1$ is subclass of training oracle in $\mathcal{T}_2$. Now consider an instantiation of the test oracle (in $\mathcal{T}_1$) $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle$. Since, M_{true} is assumed to be semantics preserving it is the case that $\phi = f$. A corresponding instantiation of test oracle (in $\mathcal{T}_2$) $EX\langle f, \mathcal{X}, \mathcal{D}\rangle$ where $\mathcal{D} = M_{true}(\mathcal{D}^+)$ is indistinguishable from $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle$ since they both draw instances from the same distribution and label them according to same function f. Hence, $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle \subset EX(f, \mathcal{X}, \mathcal{D})$ i.e. the test oracle in $\mathcal{T}_1$ is subclass of test oracle in $\mathcal{T}_2$. Hence the statement of the theorem follows from the definition of the subclass relation among learning scenarios.*

Theorem 9 *Learning using a mapping M (with errors) is reducible to Domain Adaptation. Formally, $\mathcal{T}_3 = (kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle) \subset \mathcal{T}_2 = (EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle,$ $EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle)$.*

Proof 9 *The proof is straightforward and follows from the following two observations:*
(1) $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle \subset EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle$ corresponding to $f_s = \phi$ and $\mathcal{D}_s = M(\mathcal{D}^+)$ and (2) $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle \subset EX\langle f_t, \mathcal{X}, \mathcal{D}_t$ corresponding to $f_t = f and \mathcal{D}_t = M_{true}(\mathcal{D}^+)$.

A representation of reduction between the various learning tasks is shown in Figure 6.3. These reductions can prove to be useful in practice. For example, the reduction of learning in presence of mapping errors to learning in presence of noise allows for porting of results and techniques from the latter to the former. In particular, Bshouty et al. present a PAC learning algorithm under the NSN model (refer *NastyConsistent* in Bshouty et al. (1999)). Hence, the same algorithm can be used for PAC style learning in the setting of learning in presence of mapping errors wherein the training dataset instead of being drawn from NSN oracle is now drawn from k^m-delegating oracle.

6.3.3 Choosing among multiple available mappings

Recall that, in general, it is possible to have multiple admissible as well multiple semantics preserving mappings. Hence, it is conceivable to have the following setting: learning from disparate data sources when there are two available admissible mappings M_1 and M_2. The mappings may have been developed by different groups (independent of each other) and hence differ from each other and possibly from the true mapping M_{true}. An obvious approach to deal with this scenario is to use an ensemble based approach (Rokach, 2010) where each individual classifier in the ensemble is built using a training dataset obtained by exercising a separate mapping in the set of available mappings. However, it may also be of interest to chose an appropriate mapping between M_1 and M_2 to learn in the setting of disparate data sources. This requires approach to find which of the two mappings M_1 and M_2 is a better *proxy* for the true mapping M_{true}.

Definition 12 *The degree of adaptability of an instantiated oracle $EX\langle\theta_1\rangle$ to another instantiated oracle $EX\langle\theta_2\rangle$ with respect to a learning algorithm $\mathcal{L}$ is $|R(\theta_1, \theta_2) - R(\theta_2, \theta_2)|$ and is denoted by $\delta(\theta_1, \theta_2)$.*

An oracle $EX\langle\theta_1\rangle$ is said to be **perfectly adaptable** to an oracle $EX\langle\theta_2\rangle$ if $\delta(\theta_1, \theta_2) = 0$. Intuitively the *degree of adaptability* captures the notion of how well an oracle can serve as proxy for another oracle. This notion is useful in our setting since we want to know that between the two oracles $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_1, \mathcal{X}\rangle$ and $kEX\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_2, \mathcal{X}\rangle$ which is a better proxy for the oracle $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle$. Consider the following learning scenarios:
(1) $\mathcal{T}_1 = (kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_1, \mathcal{X}\rangle, kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle)$ with the associated loss denoted by $R(\mathcal{T}_1)$ and (2) $\mathcal{T}_2 = (kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_2, \mathcal{X}\rangle, kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle)$ with the

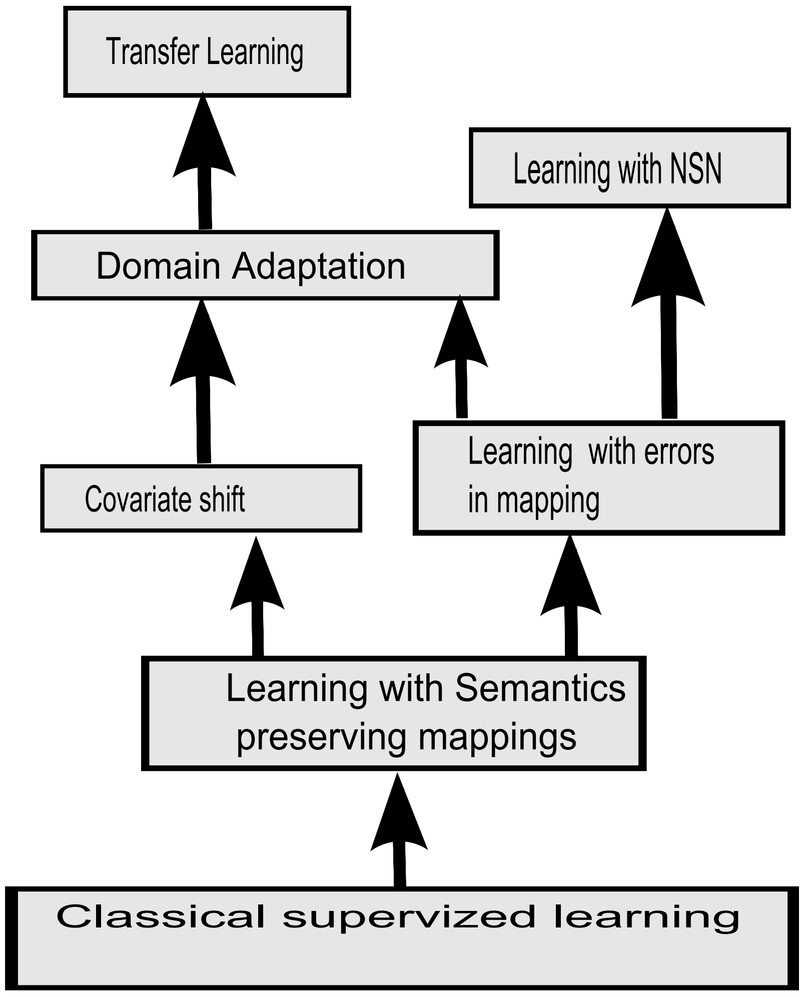
Transfer Learning
Learning with NSN
Domain Adaptation
Covariate shift
Learning with errors in mapping
Learning with Semantics preserving mappings
Classical supervized learning

Figure 6.3 A schematic representation of reduction between various learning tasks.

associated loss denoted by $R(\mathcal{T}_2)$. Define $\alpha = R(\mathcal{T}_1) - R(\mathcal{T}_2)$. Choosing among the mappings M_1 and M_2 reduces to problem of checking whether $\alpha > 0$. This can be easily done using standard statistical hypothesis testing techniques (see DeGroot (2001),Crawley (2005)). Briefly, the *null hypothesis* is $\alpha < 0$. The mapping M_1 is chosen when null hypothesis is rejected and the mapping M_2 is chosen the null hypothesis is accepted. Note that this technique does assume that some labeled test examples are available from $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle$. In general, this may not be the case (e.g. in the case of new domain where labeling instances is expensive). However, in general, it can be assumed unlabeled examples are available from the target domain (which in this case corresponds to unlabeled examples obtained from oracle $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle$). In such a case an approach to choosing between M_1 and M_2 involves checking which of the induced distributions $M_1(\mathcal{D}^+)$ and $M_2(\mathcal{D}^+)$ is *closer* to the distribution $M_{true}(\mathcal{D}^+)$. The notion of closeness between two distributions $\mathcal{D}_1$ over $\mathcal{D}_2$ is captured by means of A-distance (introduced in Kifer et al. (2004) and again in Ben-David et al. (2007)) which can be estimated from a finite number of unlabeled samples drawn from $\mathcal{D}_1$ and $\mathcal{D}_2$ (Kifer et al., 2004).

6.4 Mappings to address Covariate Shift and Domain Adaptation

In the previous section we showed that the problem of learning from disparate data using an available semantic preserving mapping which, in general, is different from the true mapping, is reducible to the problem of covariate shift which in turn is reducible to domain adaptation. As a result techniques to address the problem of covariate shift can be ported to the setting of learning from disparate data. However, now we show that mappings can be used as a tool to address covariate shift. Henceforth, unless otherwise specified we will assume that all the oracles are instantiated oracles. Consider the covariate shift learning scenario: $\mathcal{T}_1 = (EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$. Alternatively the covariate shift can be modeled by the learning scenario $\mathcal{T}_2 = (1EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$ where the subordinate oracle for the 1Delegating training oracle is $EX\langle f, \mathcal{X}, \mathcal{D}_s\rangle$ and M is an identity function. In contrast to $\mathcal{T}_1$ where the user has no control over the parameters of the oracle, in $\mathcal{T}_2$ the user can a select any mapping $M \in \mathcal{M}*$. Since the user can vary M in $\mathcal{T}_2$, let $\mathcal{T}_2(M)$ represent the learning scenario for a specific choice of M. A natural question arises that does there exist $M \in \mathcal{M}*$ such that $1EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$ is indistinguishable from $EX(f, \mathcal{X}, \mathcal{D}_t)$ in which case the covariate problem is addressed completely. We now present a result which shows that, in general, there does not exist $M \in \mathcal{M}*$ such that $1EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$ is indistinguishable from $EX(f, \mathcal{X}, \mathcal{D}_t)$.

Theorem 10 *There exists an oracle $EX\langle f, \mathcal{D}_t, \mathcal{X}\rangle$ ($|\mathcal{X}|$ is finite and greater than 1) and no mapping $M \in \mathcal{M}*$ such that 1Delegating oracle $1EX\langle \phi, \{\mathcal{X}\}, \{\mathcal{D}\}, M, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f, \mathcal{D}, \mathcal{X}\rangle$*

Proof 10 *Since $\mathcal{X}$ is finite the number of admissible mappings is finite. As a result there are a finite number distributions that can be induced over $\mathcal{X}$ by varying that mapping M in the* k*Delegating oracle $1EX\langle \phi, \{\mathcal{X}\}, \{\mathcal{D}\}, M, \mathcal{X}\rangle$. Let these distributions be $M_1(\mathcal{D}^+), M_2(\mathcal{D}^+) \ldots M_{|\mathcal{M}*|}(\mathcal{D}^+)$. Consider the following distribution over $\mathcal{X}$ ($x_1, x_2 \in \mathcal{X}$): $Pr[x_1] = r$, $Pr[x_2] = 1 - r$ and $Pr[x \notin \{x_1, x_2\}] = 0$. Since r can take any value between 0 and 1 there are infinite possible distributions over $\mathcal{X}$ when $|\mathcal{X}| > 1$. Hence, by pigeon hole principle there exists a distribution over $\mathcal{X}$ which cannot be induced by* k*Delegating oracle $1EX\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$.*

The theorem (10) shows that, in general, there does not exist a mapping that can be used to make the oracle $EX\langle f, \mathcal{D}_s, \mathcal{X}\rangle$ indistinguishable from $EX\langle f, \mathcal{D}_t, \mathcal{X}\rangle$. However, the covariate shift problem can be still addressed to a certain degree by choosing an appropriate mapping $M_1 \in \mathcal{M}*$ such that $EX1\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M_1, \mathcal{X}\rangle$ is most adaptable to $EX\langle f, \mathcal{D}_t, \mathcal{X}\rangle$. We formalize this problem as follows: find mapping $M_1 \in \mathcal{M}*$ such that $\forall\ M \in \mathcal{M}*$, $\delta(EX1\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M_1, \mathcal{X}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle) \leq \delta(EX1\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle, EX\langle f, \mathcal{X}, \mathcal{D}_t\rangle)$. An approach to solve this problem is as follows: let $\mathcal{A} = \mathcal{M}*$ be a set of candidate mappings. Repeatedly choose between any two mappings in $\mathcal{A}$ using the approach described in section 6.3.3 and at each step remove the discarded mapping from $\mathcal{A}$ until $|\mathcal{A}| = 1$. Return the final mapping in $\mathcal{A}$ as the solution. In practice, since the number of mappings is large, some user guidance may be required to restrict the candidate set of mappings $\mathcal{A}$ (instead of initializing it to the set $\mathcal{M}*$).

6.4.1 Probabilistic Mappings

The theorem (10) shows that, in general, there does not exist an attribute mapping $m^x : \mathcal{X} \mapsto \mathcal{X}$ that can be used to make the oracle $EX\langle f, \mathcal{D}_s, \mathcal{X}\rangle$ indistinguishable from $EX\langle f, \mathcal{D}_t, \mathcal{X}\rangle$. We now introduce a more general type of mapping, called probabilistic mapping, and show that there exists an probabilistic mapping that guarantees such indistinguishability and hence perfect adaptability.
Let $\Delta^n = \{(t_1 \dots t_n) \in \mathbb{R}^n | \forall i,\ t_i \geq 0\}$.

Definition 13 ***Probabilistic Mapping:*** *Given finite instance space $\mathcal{X}_S$ and $\mathcal{X}_T$, a probabilistic mapping from $\mathcal{X}_S$ to $\mathcal{X}_T$ is the function $m : \mathcal{X}_S \mapsto \Delta^{|\mathcal{X}_T|}$ where given $x \in \mathcal{X}_S$, $m(x) = \{\psi_m(x, y_1), \psi_m(x, y_2) \dots \psi_m(x, y_{|\mathcal{X}_T|})\} \in \Delta^{|\mathcal{X}_T|}$ implies that when x is sampled from $\mathcal{X}_S$ according to some distribution (say $\mathcal{D}_S$) the element returned to the learner $\mathcal{L}$ is y_i with probability $\psi_m(x, y_i)$.*

The probabilistic mapping m and the distribution $\mathcal{D}_S$ is said to induce a distribution $m(\mathcal{D}_S)$ over the instance space $\mathcal{X}_T$. A mapping $M^x : \mathcal{X}_S \mapsto \Delta^{|\mathcal{X}_T|}$ is said to be *admissible* iff $\forall x \in \mathcal{X}_S, \sum_{y \in \mathcal{X}_T} \Psi(x, y) = 1$.
Observation An admissible probabilistic mapping ensures that given an element is sampled from $\mathcal{X}_S$ according to a valid probability distribution (say $\mathcal{D}_s$), the induced distribution (say $M(\mathcal{D}_s)$) with which an instance is returned from $\mathcal{X}_T$ to $\mathcal{L}$ is also a valid probability distribution.

We now introduce a k-Delegating oracle with probabilistic mappings. Let $EX^1\langle f_1, \mathcal{X}_{s^1}, \mathcal{D}_1\rangle \dots EX^k\langle f_k, \mathcal{X}_{s^k}, \mathcal{D}_k\rangle$ be k subordinate oracles. A *probabilistic mapping k-delegating oracle* has access to a mapping set $M = \{m_1, m_2 \dots m_k\}$ where $m_i = \{m_i^x, m_i^c\}$; $m_i^x : \mathcal{X}_{s^i} \longrightarrow \Delta^{|\mathcal{X}|}$ is an admissible *probabilistic mapping function* (called attribute mapping function); and $m_i^c : C_i \longrightarrow C$ is a *class mapping function* where $C_i = Range(f_i)$ and $C = Range(f)$. It invokes subordinate oracles $EX^1\langle f_1, \mathcal{X}_{s^1}, \mathcal{D}_1\rangle \dots EX^k\langle f_k, \mathcal{X}_{s^k}, \mathcal{D}_k\rangle$ where the ith subordinate oracle $EX^i\langle f_i, \mathcal{X}_{s^i}, \mathcal{D}_i\rangle$ returns examples of the form $\langle x_{s^i}, f_i(x_{s^i})\rangle$ where x_{s^i} is drawn from X_{s^i}

according to $\mathcal{D}_i$ and $f_i \in \mathcal{F}^i$. The labeled example returned to the learner is $\langle y, m_i^c(f_i(x_{s^i}))\rangle$ with probability $\psi_{m_i^x}(x_{s^i}, y)$ (where $y \in \mathcal{X}$). The mapping m_i is said to be *semantics preserving* if $\forall x_{s^i} \in \mathcal{X}_{s^i}\ \psi_{m_i^x}(x_{s^i}, y)! = 0$ implies $f(y) = m_i^c(f_i(x_{s^i}))$. The mapping set M is said to be semantics preserving if $\forall m \in M,\ m$ is semantics preserving. Intuitively, a semantics preserving mapping ensures that the examples returned to the learner are labeled according to the target function f. A probabilistic mapping oracle is denoted by $EX1\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M, \mathcal{X}\rangle$ where ϕ is the labeling function and and M is a probabilistic mapping set.

Consider a distribution $\mathcal{D}_s$ over the instance space $\mathcal{X}_s$. The distribution $\mathcal{D}_s$ can be seen as the function $\mathcal{D}_s : \mathcal{X} \mapsto \mathbb{R}^+$ that associates with each instance $x \in \mathcal{X}_s$ a positive real number, denoted by $w_{\mathcal{D}_s}(x)$, called the weight of the instance x in the distribution $\mathcal{D}_s$. The distribution $\mathcal{D}_s$ is a probability distribution if $\sum_{x \in \mathcal{X}_s} w_{\mathcal{D}_s}(x) = 1$.

Lemma 11 *Given a fixed distribution $\mathcal{D}_s$ over the finite instance space $\mathcal{X}_s$ and a possible distribution $\mathcal{D}_t$ over the finite instance space $\mathcal{X}_t$ such that $\sum_{x \in \mathcal{X}_s} w_{\mathcal{D}_s}(x) = \sum_{x \in \mathcal{X}_t} w_{\mathcal{D}_t}(x)$, there exists a probabilistic mapping $m^x : \mathcal{X}_s \mapsto \Delta^{|\mathcal{X}_t|}$ such that $m^x(\mathcal{D}_s)$, the distribution induced by m^x and $\mathcal{D}_s$ over $\mathcal{X}_t$ is the same as the distribution $\mathcal{D}_t$.*

Proof 11 *See Appendix A*

Theorem 12 *Let $\mathcal{X}_s$ and $\mathcal{X}_t$ be finite instances spaces. Given oracles $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$ and $EX(f_s, \mathcal{X}_s, \mathcal{D}_s)$ where $f_t : \mathcal{X}_t \mapsto \{0,1\}$ and $f_s : \mathcal{X}_s \mapsto \{0,1\}$ and $\sum_{x \in \mathcal{X}_s^+} Pr_{x \in \mathcal{D}_s}[x] = \sum_{x \in \mathcal{X}_t^+} Pr_{x \in \mathcal{D}t}[x]$ there exist a probabilistic mapping $M = \{m^x, m^c\}$ where $m^x : \mathcal{X}_s \mapsto \Delta^{|\mathcal{X}_t|}$ and $m^c : Range(f_s) \mapsto Range(f_t)$ such that the 1Delegating oracle $1EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$.*

Proof 12 *Intuitively, the proof uses Lemma 11 to construct a probabilistic attribute mapping between $\mathcal{X}_s^+$ and $\mathcal{X}_t^+$ (i.e. instances with label 1) and again a probabilistic attribute mapping between $\mathcal{X}_s^-$ and $\mathcal{X}_t^-$ (i.e instances with label 0). It then combines these mappings to construct a required semantics preserving probabilistic mapping M that ensures that 1Delegating oracle $1EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$. For complete details of proof see Appendix B.*

The theorem 12 shows that there exists a probabilistic mapping that solves the domain adaptation problem $\mathcal{T}_1 = (EX\langle f_s, \mathcal{X}, \mathcal{D}_s\rangle, EX\langle f_t, \mathcal{X}, \mathcal{D}_t\rangle)$ when it is the case that $\sum_{x \in \mathcal{X}_s^+} Pr_{x \in \mathcal{D}_s}[x] = \sum_{x \in \mathcal{X}_t^+} Pr_{x \in \mathcal{D}_t}[x]$. However, the requirement that $\sum_{x \in \mathcal{X}_s^+} Pr_{x \in \mathcal{D}_s}[x] = \sum_{x \in \mathcal{X}_t^+} Pr_{x \in \mathcal{D}_t}[x]$ is a strong constraint and, in practice, will often not be satisfied. To solve $\mathcal{T}_1$, in the most general sense, we introduce an another type of the probabilistic mapping called *probabilistic instance mapping* that maps a labeled instances drawn from a subordinate oracle to a labeled instance that is returned to the learner. In contrast the earlier introduce mappings have two components: (1) the attribute mapping function and (2) the class mapping function, that are used in conjunction to convert a labeled instance from the subordinate oracle to a labeled instance that is passed to the learner.

A k-Delegating oracle with *probabilistic instance mappings* is a k-delegating oracle which has access to a mapping set $M^x = \{m_1^x, m_2^x \dots m_k^x\}$ where $m_i^x : \mathcal{X}_{s^i \times Range(f_i)} \mapsto \Delta^{|\mathcal{X}|+1}$ is an admissible *probabilistic mapping function* and given the selected subordinate oracle samples the labeled example $\langle x_{s^i}, f_i(x_s^i)\rangle$, the labeled exampled returned to $\mathcal{L}$ is $y \in \mathcal{X}$ with probability $\psi(\langle x_{s^i}, f_i(x_{s^i})\rangle, y)$. We denote k-Delegating oracle with instance mappings by $EX1\langle \phi, \mathcal{X}^+, \mathcal{D}^+, M^x, \mathcal{X}\rangle$ where ϕ is the labeling function and M^x is a probabilistic instance mapping set. Note that we use M^x to denote a probabilistic instance mapping set as opposed to M that we used to denote probabilistic mapping set.

Theorem 13 *Let $\mathcal{X}_s$ and $\mathcal{X}_t$ be finite instances spaces. Given oracles $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$ and $EX\langle f_s, \mathcal{X}_s, \mathcal{D}_s\rangle$ where $f_t : \mathcal{X}_t \mapsto \{0,1\}$ and $f_s : \mathcal{X}_s \mapsto \{0,1\}$ there exist a probabilistic instance mapping $M^x = \{m^x\}$ where $m^x : \mathcal{X}_s \times \{0,1\} \mapsto \Delta^{|\mathcal{X}_t|+1}$ such that the 1Delegating oracle $1EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M^x, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$.*

Proof 13 *The distribution $\mathcal{D}_s$ and function f_s induce a distribution over $\mathcal{X}_s \times \{0,1\}$ the instance space of the labeled examples drawn from $EX(f_s, \mathcal{X}_s, \mathcal{D}_s)$. Let this distribution be denoted by $\mathcal{D}_t^{f_s}$. Similarly, let $\mathcal{D}_t^{f_t}$ be the distribution induced over $\mathcal{X}_t \times \{0,1\}$ by $\mathcal{D}_t$ and function f_t. By Lemma 11 there exists a probabilistic mapping $M^x : \mathcal{X}_s \times \{0,1\} \mapsto \Delta^{|\mathcal{X}_s|+1}$ such that M^x and $\mathcal{D}_t^{f_s}$ induce $\mathcal{D}_t^{f_t}$. Hence, $EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$.*

A straightforward implication of Theorem 13 (for the case $\mathcal{X}_s = \mathcal{X}_t$) is that there exists a probabilistic instance mapping that solves the domain adaptation problem. Consequentially, it is theoretically possible to learn about a given domain using training examples available from some other domain. However, the Theorem 13 is an existence result and finding such an probabilistic mapping is an *open problem.* However, it turns out that some known approaches to achieve domain adaptation can be viewed in terms of probabilistic mappings. Consider approaches to correcting sample bias and domain adaptation that involve reweighing of instances (Cortes et al., 2008; Jiang and Zhai, 2007). In general, reweighing can be seen as means of changing the underlying distribution. Hence, it can accomplished in a straightforward manner by using an appropriate probabilistic mapping. Reweighing of instances is also a critical component in AdaBoost (Freund and Schapire, 1997) where the weights associated with each instances in the dataset are increased or decreased (for the next iteration) based of whether they were correctly or incorrectly classified (in the current iteration) by a base classifier. Hence, AdaBoost can be seen as an ensemble classifier each with a different probabilistic mapping. A variation of AdaBoost has been used to address domain adaptation in Dai et al. (2007). The use of AdaBoost for domain adaptation is hardly surprising in the light that it can be explained in terms of probabilistic mappings and that probabilistic mappings can be used to solve domain adaptation.

6.5 Summary and Related Work

There is growing interest in the problem of learning predictive models from distributed data sources [Park and Kargupta (2003); Caragea (2004)]. Caragea et al. (2005) have described algorithms that provide rigorous performance guarantees (relative to their single data source counterparts) for learning from distributed, semantically disparate data sources when the mappings are semantics preserving. Crammer et al. (2008) have examined the problem of learning predictors from a set of *related* data sources. Ben-david et al. (2002) have analyzed the sample complexity of learning from semantically disparate data sources in a setting where classifiers trained on data sources $D_1 \cdots D_{n-1}$ are used to predict the class labels of instances from a data source D_n. There has been very little work on detecting mapping errors in the setting of learning from disparate data sources. Of related interest is the work in ontology mapping field [Kalfoglou and Schorlemmer (2005); Euzenat and Shvaiko (2007)]. However, the primary focus in this area is aligning ontologies (through use of mappings), merging related ontologies or detecting logical inconsistencies in mappings [Meilicke et al. (2007)]. In contrast, the focus of our work is on the challenges that arise in learning from disparate data sources once a mapping or a set of mappings is available.

Domain Adaptation and Transfer Learning have been studied extensively in literature with the terms often being used interchangeably (see Pan (2010) for a comprehensive list of papers in this field). Pan and Yang (2010) provide a good survey on Transfer Learning. Domain Adaptation has been used extensively in a variety of applications including text classification and sentiment analysis (Jiang, 2008; Blitzer et al., 2007b). Techniques to apply transfer learning in novel domains (besides text classification) are presented in Yang (2009). A statistical formulation of the Domain Adaptation problem and several straightforward techniques to solve it to address the domain adaptation are presented by Daumé III (2007) Learning bounds for Domain Adaptation have been studied in detail in literature [Blitzer et al. (2007a); Ben-David et al. (2010); Mansour et al. (2009)]. Ben-David et al. (2007) show that a good feature representation is a crucial factor in the success of domain adaptation. Mansour et al. (2008) present a theoretical analysis of the problem of domain adaptation with multiple sources. Reduction between learning tasks have been used previously in other contexts [Kearns and Vazirani (1994)]. The work that closely resembles ours is the notion of error limiting reductions introduced in Beygelzimer et al. (2005). However, except in Koul and Honavar (2010) which relates mapping errors to noise, there has been no work (to the best of our knowledge) in reducing the problems in the setting of learning from disparate data sources to some aspect of supervised learning.

In this chapter we introduced the notion of a learning scenario to model some well know class of supervised problems. In addition, we introduce a notion of reducibility among classes of supervised learning tasks and showed that several aspects learning from semantically disparate data sources can be reduced to, and hence understood in terms of the theoretically well-studied problem of domain adaptation. Furthermore, we showed that there exists a probabilistic instance mapping that facilitates domain adaptation. To the best of our knowledge there is no work that aims to relate aspects of learning from semantically disparate data sources to domain adaptation. This analysis has lead some interesting open problems in this setting. On application side, it is an open problem to explore approaches to compute the probabilistic mapping (or an good approximation in a PAC like setting) that facilitates domain adaptation from a sample of instances available from the two domains. On the theory side it is an open problem to figure out that does a probabilistic mapping (that facilitates domain adaptation) exist between two domains with infinite instance spaces (and laying out any additional assumptions that may be required in this setting). Further, there are several interesting directions along which the analysis presented in this chapter can be extended. We want to describe (and compute) an appropriate notion of closeness between mappings that can be used to bound the error of using a semantics preserving mapping as a proxy for the true mappings (see Appendix C where we describe the notion of μ closeness between mappings and use it an error bound that inspired by similar bound for domain adaptation in Blitzer et al. (2007a)). In addition, we want study the effect of mapping errors in multi-relational learning and multiple instance learning.

CHAPTER 7. SUMMARY AND CONTRIBUTIONS

7.1 Thesis Summary

Machine learning approaches offer some of the most successful techniques for constructing predictive models from data. However, applying such techniques in practice requires overcoming several challenges: infeasibility of centralized access to the data because of the massive size of some of the data sets that often exceeds the size of memory available to the learner, distributed nature of data, data fragmentation, access restrictions and data sources that evolve spatially, temporally, or spatio-temporally (e.g. data streams and genomic data sources in which new data is being submitted continuously). Further, often data about related domains is collected by independent entities in the context of the problem they are addressing and as such the resultant data sources differ not only in structure and organization but also in the semantics associated with the data. In the Semantic Web vision this corresponds to autonomous data sources using similar but different ontologies to associate meaning with data. Hence, this setting requires the development of techniques and algorithms to learn in presence of semantic disparity introduced due the use of different but related ontologies by data sources of interest to a learner. Additional challenges in this setting include the ability of learners to cope with errors in the mappings that are used to resolve semantic disparity and the ability to select among multiple available mappings. Consequentially, there is a pressing need for learning algorithms that are scalable, don't assume direct access to data, are able to cope with frequent data updates and are able to handle data fragmentation and semantic disparity.

In this dissertation we show that learning from datasets using statistical queries and semantic correspondences that present a unified view of disparate data sources to the learner offer a powerful general framework for addressing some of the challenges that occur in the setting of learning from disparate data sources. Building on the work of Caragea et al. to learn decision trees from horizontally distributed data sources we precisely described the SQL count queries required to build Naive Bayes and Decision Trees from datasets stored in Relational Database Management Systems . We described approaches to minimize the number of queries submitted to a database and show that imputation approaches to handling missing values can be applied in setting of learning using statistical queries by addressing their effect on the statistics required to build Naive Bayes and decision trees. Our analysis showed that sufficient statistics based approach allowed us to cope with massive data size since instead of loading the entire dataset in the memory, it is only required to load the answers to the required statistical queries. We extended the sufficient statistics based approach to the setting of learning and updating Markov Property based predictors for sequence classification. We described a planner to answer count queries from semantically disparate data sources that are fragmented (horizontally and/or vertically) from a user point of view. This planner in conjunction with the approach of learning from data using statistical queries allows for knowledge acquisition from semantically disparate data.

We introduced an extension to classical oracle based model of supervised learning to model learning from disparate data sources and used it to demonstrate the theoretical equivalence of a certain class of inter-ontology mapping errors and noise models, and hence the problem of learning in the presence of mapping errors from semantically disparate data to the problem of learning from noisy data. Furthermore, we showed that several aspects learning from semantically disparate data sources such as how to choose an optimal mapping from among a set of alternative expert-supplied or automatically generated mappings can be reduced to, and hence understood in terms of the theoretically well-studied problem of domain adaptation. Finally, we introduced the notion of *probabilistic mappings* and showed that there exists a specific probabilistic mapping that facilitates domain adaptation.

7.2 Contributions

This thesis focused on several aspects of the problem of learning predictive models from semantically disparate data sources. The key elements of our approach to this problem include: (1) Taking advantage of the general strategy for transforming a broad class of standard learning algorithms that assume in memory access to a dataset into algorithms that interact with the data source(s) only through statistical queries, (2) using a data integration system that presents a unified view of disparate data sources to the learner and (3) reduction of several problems that occur in the setting of learning from disparate data (e.g. learning in presence of mapping errors) to some previously studied problem in literature. The main contributions of this thesis include:

- Development of approaches to deal with missing values in the statistical query based algorithms for building decision trees and the techniques to minimize the number of queries in such a setting.
- Development of planner to answer count queries from semantically disparate data sources that are fragmented (horizontally and/or vertically) from a user point of view.
- Development and open-source implementation of an ontology-based system for querying multiple semantically disparate data sources from a user's point of view.
- Development of sufficient statistics based algorithms for constructing and updating sequence classifiers.
- Demonstration of the theoretical equivalence of a certain class of inter-ontology mapping errors and noise models, and hence the reduction of the problem of learning in the presence of mapping errors from semantically disparate data to the problem of learning from noisy data.
- Reduction of several aspects of learning from semantically disparate data sources (such as (a) how errors in mappings affect the accuracy of the learned model and (b) how to choose an optimal mapping from among a set of alternative expert-supplied or automatically

generated mappings) to the well studied problem of domain adaptation (i.e., adapting a model that is trained on data sampled according to a distribution that is different from the distribution from which the test data are obtained).

7.2.1 Published Work

The work carried out during this thesis resulted in the following publications:

- **Learning Classifiers from Large Databases Using Statistical Queries (Koul et al. (2008))**. In this paper, published in *Web Intelligence 2008*, we describe a framework to learn Naive Bayes and decision trees predictors from datasets stored in Relational Database Management Systems (such as mySQL) using SQL count queries. In such a setting, we describe techniques for handling missing values in the dataset without the need to access the underlying dataset or the execution of the user defined code in the data repositories. We described the the effect of missing values in the dataset on the number of queries required to build the Naive Bayes and decision tree classifier and outlined some optimizations techniques to minimize the number of queries required.

- **Design and implementation of a Query Planner for Data Integration (Koul and Honavar (2009))**. In this paper, published in *ICTAI 2009*, we present a query planner that allows a user to answer statistical queries from a set of distributed data sources addressing the twin problems of data fragmentation (horizontal and/or vertical) and semantic disparity (schema heterogeneity as well data content heterogeneity).

- **Scalable, Updatable Predictive Models for Sequence Data (Koul et al. (2010))**. In this paper, published in *BIBM 2010*, we describe an approach to learn from massive sequence data sets using statistical queries. Specifically we show how Markov Models and Probabilistic Suffix Trees(PSTs) can be constructed from databases that answer specific count queries. We analyze the query complexity (a measure of the number of queries needed) for constructing classifiers in such settings and outline some techniques to minimize the query complexity. In addition, we described how Markov model based predictors can be updated in response to addition or deletion of subsets of the data.

- **Learning in Presence of Ontology Mapping Errors (Koul and Honavar (2010))** In this paper, published in *Web Intelligence 2010*, we address the problem on learning from disparate data in the setting when the available ontology mappings used to resolve semantic disparity have errors. We introduce an extension to the classical oracle based model of learning that allows us to model learning from disparate data sources. We use this extended model to show that learning from semantically disparate data sources in the presence of mapping errors can be reduced to the problem of learning from a single data source in the presence of nasty classification noise within a PAC-like framework. This reduction, of learning in the presence of mapping errors to learning in the presence of nasty classification noise, opens up the possibility of applying existing results and approaches to learning in presence of classification noise to the problem of learning in the presence of mapping errors.

- **On the Relationship between Learning Classifiers from Semantically Disparate Data and Domain Adaptation (to be submitted)**. In this paper we introduce the notion of a learning scenario to model several well studied problems in supervised learning such as Learning from Disparate Data, Learning under Covariate Shift, Domain Adaptation and Transfer Learning. In addition, we introduce a notion of reducibility among classes of supervised learning tasks and showed that several aspects learning from semantically disparate data sources such as (a) how errors in mappings affect the accuracy of the learned model and (b) how to choose an optimal mapping from among a set of alternative expert-supplied or automatically generated mappings can be reduced to, and hence understood in terms of the theoretically well-studied problem of domain adaptation. Furthermore, we introduce the notion of probabilistic mappings and show that there exists a specific probabilistic mapping that facilitates domain adaptation (i.e., adapting a model that is trained on data sampled according to a distribution that is different from the distribution from which the test data are obtained)

7.2.1.1 Published work not included in the thesis

Other published work not included in the thesis are:

- **Identifying and Eliminating Inconsistencies in Mappings across Hierarchical Ontologies**. In this paper, published in *ODBASE 2010*, we consider the problem of identifying the largest consistent subset of mappings in the restricted, yet practically important setting of hierarchical ontologies. Specifically, we consider mappings that assert that a concept in one ontology is a subconcept, superconcept, or equivalent concept of a concept in another ontology. We show that even in this simple setting the task of identifying the largest consistent subset is NP-hard. We explore several polynomial time algorithms for finding suboptimal solutions including a heuristic algorithm to this problem. We present results of experiments using several synthetic as well as real-world ontologies and mappings that demonstrate the usefulness of the proposed algorithm.

- **Complexes of On-Line Self Assembly**. The Tile Assembly Model (TAM) is an abstract mathematical model of nanoscale self-assembly [Rothemund (2001)]. In this paper, published in *EIT 2008*, we introduced variations of the TAM called Fair On-line Assembly (FOAF) and explored certain properties that a TAM must possess in order for it to be a FOAF.

- **ANEXdb:An integrated Animal aNnotation and microarray EXpression Database**. This paper, published in *Mammalian Genome 2009*, describes an open-source web application that supports integrated access of two databases that house microarray expression and EST(Expressed sequence tag) annotation data . The web application is currently available at `http://www.anexdb.org`.

The thesis resulted in the following open source software.

- **Indus Learning Framework**:

  A suite of machine learning algorithms that learn from datasets using sufficient statistics. The current implementation includes Naive Bayes and Decision Tree classifiers, and can be extended to incorporate more classifiers that are amenable to the sufficient statistics approach. The code is open sourced at `http://code.google.com/p/induslearningframework/`.

- **Indus Integration Framework**:

  A data integration system that allows a user to pose count queries over a collection of physically distributed, autonomous, semantically heterogeneous data sources as though they were a collection of tables structured according to an ontology supplied by the user. The framework is extensible to use multiple ontology formats (at present OWL and custom format supported), data source types (e.g. RDBMS , ARFF files, Web Services) and reasoners (at present Pellet and internal reasoner supported). The framework in conjunction with *Indus Learning framework* allows for knowledge acquisition from semantically disparate data sources. The code is open sourced at `http://code.google.com/p/indusintegrationframework` .

7.3 Future Work

Some promising directions for future work include:

1. *Extension of Learning using Statistical Queries Paradigm to RDF Data.* The Resource Description Framework (RDF) is a language for representing information about resources in the World Wide Web [Manola and Miller (2004)]. The rise of Semantic Web has resulted in increasing availability of data in RDF format. Even data in relational databases is being made available in RDF format as tools for publishing relational databases on the Semantic Web become available (see W3C (2010a)). For example, the DBLP bibliography database is now available in RDF format (see DBLP (2010)). Learning from RDF data presents challenges similar to those that have been discussed in this thesis. These challenges include a learner's ability to handle massive data size (e.g. Tauberer (2010) lists a census dataset with one billion RDF Triples). Learning from RDF data using statistical queries provides a straightforward approach to handle massive data size. Most RDF data source provide the ability to query the RDF data using SPARQL (the

standard query language for RDF data). Though the first version of RDF data did not contain support for count queries, the current version (SPARQL 1.1 [W3C (2010b)]) does include support for aggregate operations. This opens up the possibility of extending the approach of learning using statistical queries to RDF data. We believe it should be possible to learn Relational Bayesian Classifiers [Neville et al. (2003)] from RDF data using SPARQL 1.1 aggregate queries in a straightforward manner.

2. *Incorporate more Expressive Mappings to the Setting of Learning from Disparate Data.* The approach to learning from disparate data sources discussed in this thesis is restricted to the case when attribute mappings are one to one. Hence, the possible ontology mappings are restricted to those between concepts in the ontologies associated with the mapped attributes. An interesting direction for future work would be to extend the described approach (of learning from semantically disparate data sources) to handle more expressive schema and ontology mappings such as the case when an attribute in a user view can be mapped to two or more attributes in the data source view. In such as case, a concept in an ontology associated with an attribute in user view can be composed from concepts in multiple ontologies (each of which may be associated with an unique attribute in the datasource view). For example, a concept $AdultMale$ in an ontology associated with an attribute in user view could be composed as an intersection of the $Male$ Concept (say present in an ontology associated with attribute Sex) and the $Adult$ concept (say associated with attribute Age).

3. *Incorporating Ontology Evolution* Ontologies evolve over time resulting in different versions of the same ontology being available [Shaban-Nejad and Haarslev (2009), Flouris et al. (2008)]. An interesting direction for future research could to be explore learning predictive models in presence of evolving ontologies and mappings without needing to rebuild the predicted models from scratch.

APPENDIX A.

Lemma Given a fixed distribution $\mathcal{D}_s$ over the finite instance space $\mathcal{X}_s$ and a possible distribution $\mathcal{D}_t$ over the finite instance space $\mathcal{X}_t$ such that $\sum_{x\in\mathcal{X}_s} w_{\mathcal{D}_s}(x) = \sum_{x\in\mathcal{X}_t} w_{\mathcal{D}_t}(x)$, there exists a probabilistic mapping $m^x : \mathcal{X}_s \mapsto \Delta^{|\mathcal{X}_t|}$ such that $m^x(\mathcal{D}_s)$, the distribution induced by m^x and $\mathcal{D}_s$ over $\mathcal{X}_t$ is the same as the distribution $\mathcal{D}_t$.

Proof 14 *Given the distribution $\mathcal{D}_s$ over $\mathcal{X}_s$, let $Pr_{x\in\mathcal{D}_s}[x] = w_{\mathcal{D}_s}(x)$. Recall $w_{\mathcal{D}_s}(x)$ is the weight of instance x in distribution $\mathcal{D}_s$. Consider a possible distribution $\mathcal{D}_t$ over $\mathcal{X}_t$ such that $Pr_{y\in\mathcal{D}_t}[y] = w_{\mathcal{D}_t}(y)$. Since $\mathcal{X}_s$ and $\mathcal{X}_t$ are finite it is possible to arrange the elements in increasing order of their weights in distribution $\mathcal{D}_s$ and $\mathcal{D}_t$. WLOG let $\mathcal{X}_s = \{x_1, x_2 \ldots x_{|\mathcal{X}_s|}\}$ be such an ordering. Similarly, let $\mathcal{X}_t = \{y_1, y_2 \ldots y_{|\mathcal{X}_t|}\}$ be an ordering of elements in $\mathcal{X}_t$ in increasing order of their weights in distribution $\mathcal{D}_t$. The statement of the lemma exists if there exists a probabilistic mapping m^x such that*

$$Pr_{y\in\mathcal{D}_t}[y] = Pr_{y\in m^x(\mathcal{D}_s)}[y] \tag{A.1}$$

Given that for an $x \in \mathcal{X}_s, m^x(x) = \{\psi_{m^x}(x, y_1), \psi_{m^x}(x, y_2) \ldots \psi_{m^x}(x, y_{|\mathcal{X}_T|})\}$ the equation (A.1) is equivalent to

$$w_{\mathcal{D}_t}(y) = \sum_{x\in\mathcal{X}_s} \psi^x_m(x, y) w_{\mathcal{D}_s}(x) \tag{A.2}$$

Hence, the lemma holds if there exists an mapping m such that equation (A.2) holds. The Algorithm 3 describes a procedure to construct a mapping for which equation (A.2) is true.

An straightforward analysis of the construction procedure shows that the constructed mapping m^x indeed satisfies equation (A.2). Intuitively, construction involves taking an instance in $\mathcal{X}_t$ one at a time and finding the set of instances in $\mathcal{X}_s$ (taken in increasing order of their weights) whose weight equals or exceeds the weight (under distribution $\mathcal{D}_t$) of chosen instance

Algorithm 3: Procedure to construct a probabilistic mapping that satisfies equation (A.2)

```
Data: Require $\mathcal{X}_s = \{x_1, x_2 \ldots x_{|\mathcal{X}_s|}\}$ to be an ordering of elements in $\mathcal{X}_t$ in increasing
      order of their weights in distribution $\mathcal{D}_s$ and (2) $\mathcal{X}_t = \{y_1, y_2 \ldots y_{|\mathcal{X}_t|}\}$ to be an
      ordering of elements in $\mathcal{X}_t$ in increasing order of their weights in distribution $\mathcal{D}_t$.
Result: Mapping $m^x$ that satisfies equation (A.2)
Initialize $m^x$ as: $\forall q \in \{1, 2 \ldots |\mathcal{X}_s|\}$ and $\forall r \in \{1, 2 \ldots |\mathcal{X}_t|\}$ set $\psi_{m^x}(x_q, y_r) = 0$ ;
Initialize variable $i_s = 1$ and $i_t = 1$ ;
while $i_t \leq |\mathcal{X}_t|$ do
    Find least $j \geq i_s$ such that $\sum_{p=i_s}^{j} w_{\mathcal{D}_s}(x_p) \geq w_{\mathcal{D}_t}(y_{i_t})$ ;
    if $j > i_s$ then
        $partial = \frac{w_{\mathcal{D}_t}(y_{i_t}) - \sum_{p=i_s}^{j-1} w_{\mathcal{D}_s}(x_p)}{w_{\mathcal{D}_s}(x_j)}$
    else The weight of instances between $i_s$ and $j$ exactly matches
        $partial = \frac{w_{\mathcal{D}_t}(y_{i_t})}{w_{\mathcal{D}_s}(x_j)}$ ;
    for $p = i_s; p < j; p++$ do
        $\psi_{m^x}(x_p, y_{i_t}) = 1$ ;
    $\psi_m(x_j, y_{i_t}) = partial$ ;
    Update $w_{\mathcal{D}_s}(x_j) = (1 - partial) w_{\mathcal{D}_s}(x_j)$ ;
    if $partial == 1$ then
        $i_s = j + 1$ ;
    else
        $i_s = j$ ;
    $i_t = i_t + 1$ ;
```

in $\mathcal{X}_t$. Then the entire weight of each instance in this chosen subset of $\mathcal{X}_s$ (except the last one with has the greatest weight) is assigned to the chosen instance in $\mathcal{X}_t$. An appropriate amount of weight (corresponding to value of partial in the algorithm) of the last instance in the chosen subset is assigned to the element in $\mathcal{X}_t$. This process is repeated for rest of instances in $\mathcal{X}_t$ (one at a time) while the possible choice of subset of $\mathcal{X}_s$ is restricted to those elements that have not yet been selected or only have part of their weight assigned in the previous step. The procedure is guaranteed to work for each *instance in $\mathcal{X}_t$ since $\sum_{x \in \mathcal{X}_s} w_{\mathcal{D}_s}(x) = \sum_{x \in \mathcal{X}_t} w_{\mathcal{D}_t}(x)$.*

APPENDIX B.

Theorem Let $\mathcal{X}_s$ and $\mathcal{X}_t$ be finite instances spaces. Given oracles $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$ and $EX(f_s, \mathcal{X}_s, \mathcal{D}_s)$ where $f_t : \mathcal{X}_t \mapsto \{0,1\}$ and $f_s : \mathcal{X}_s \mapsto \{0,1\}$ and $\sum_{x\in\mathcal{X}_s^+} Pr_{x\in\mathcal{D}_s}[x] = \sum_{x\in\mathcal{X}_t^+} Pr_{x\in\mathcal{D}t}[x]$ there exist a probabilistic mapping $M = \{m^x, m^c\}$ where $m^x : \mathcal{X}_s \mapsto \Delta^{|\mathcal{X}_t|}$ and $m^c : Range(f_s) \mapsto Range(f_t)$ such that the *1*Delegating oracle $1EX\langle\phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$.

Proof 15 *Intuitively, the proof uses Lemma 11 to construct a probabilistic mapping between $\mathcal{X}_s^+$ and $\mathcal{X}_t^+$ (i.e. instances with label 1) and again a probabilistic mapping between $\mathcal{X}_s^-$ and $\mathcal{X}_t^-$ (i.e instances with label 0). It then combines these mappings to construct a required semantics preserving probabilistic mapping M that ensures that* 1*Delegating oracle $1EX\langle\phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X}\rangle$ is indistinguishable from $EX\langle f_t, \mathcal{X}_t, \mathcal{D}_t\rangle$*

Given $\mathcal{D}_s$ over $\mathcal{X}_s$, let $\mathcal{D}_s^+$ and $\mathcal{D}_s^-$ be the resulting distribution over $\mathcal{X}_s^+$ and $\mathcal{X}_s^-$ respectively. Similarly, let $\mathcal{D}_t^+$ and $\mathcal{D}_t^-$ denote the distribution over $\mathcal{X}_t^+$ and $\mathcal{X}_t^-$ respectively. By Lemma 11 there exists a probabilistic mapping $m_+ : \mathcal{X}_s^+ \mapsto \Delta^{|\mathcal{X}_t^+|}$ such that m_+ and $\mathcal{D}_s^+$ induce $\mathcal{D}_t^+$. Again by Lemma 11 there exists a probabilistic mapping $m_- : \mathcal{X}_s^- \mapsto \Delta^{|\mathcal{X}_t^-|}$ such that m and $\mathcal{D}_s^-$ induce $\mathcal{D}_t^-$. WLOG let $\mathcal{X}_t^+ = \{y_1, y_2 \dots y_{|\mathcal{X}_t^+|}\}$ and $\mathcal{X}_t^- = \{y_{|\mathcal{X}_t^+|+1}, \dots y_{|\mathcal{X}_t|}\}$. Given $x \in \mathcal{X}_s^+, m_+(x) = \{\psi_{m_+}(x, y_1) \dots \psi_{m_+}(x, y_{|\mathcal{X}_s^+|})\}$ and given $x \in \mathcal{X}_s^-, m_-(x) = \{\psi_{m_+}(x, y_{|\mathcal{X}_t^+|+1}) \dots \psi_{m_+}(x, y_{|\mathcal{X}_t|})\}$. Now consider a mapping $m^x : \mathcal{X}_s \mapsto \Delta^{|\mathcal{X}_t|}$ constructed in the following way:

$$\forall x \in \mathcal{X}_s^+, \psi_{m^x}(x, y_i) = \begin{cases} \psi_{m_+}(x, y_i) & \textit{if } y_i \in \mathcal{X}_t^+ \\ \textit{0} & \textit{if } y_i \in \mathcal{X}_t^- \end{cases}$$

$$\forall x \in \mathcal{X}_s^-, \psi_{m^x}(x, y_i) = \begin{cases} \psi_{m_-}(x, y_i) & \textit{if } y_i \in \mathcal{X}_t^- \\ \textit{0} & \textit{if } y_i \in \mathcal{X}_t^+ \end{cases}$$

The class mapping function m^c is trivial and is assumed to map the label zero to the label zero and the label one to label one. By construction when $x \in \mathcal{X}_s^+, \psi_{m^x}(x, y) = 0$ for all $y \in \mathcal{X}_t^-$. Similarly, for $x \in \mathcal{X}_s^-, \psi_{m^x}(x, y) = 0$ for all $y \in \mathcal{X}_t^+$. As a result the probability that an instance with incorrect label is passed to the learner is zero and hence $M = \{m^x, m^c\}$ is semantics preserving. Consider the case when $y \in \mathcal{X}_t^+$. It follows that $Pr_{y \in m^x(\mathcal{D}_s)}[y] = \sum_{x \in \mathcal{X}_s} \psi_{m^x}(x, y) = \sum_{x \in \mathcal{X}_s^+} \psi_{m^x}(x, y) = \sum_{x \in \mathcal{X}_s^+} \psi_{m_+}(x, y)$ (we used the following facts: (1) $\psi_{m^x}(x, y) = 0 \ \forall x \in \mathcal{X}_s^-$ when $y \in \mathcal{X}_t^+$; and (2) $\sum_{x \in \mathcal{X}_s^+} \psi_{m^x}(x, y) = \sum_{x \in \mathcal{X}_s^+} \psi_{m_+}(x, y)$ when $y \in \mathcal{X}_t^+$).

Again consider $y \in \mathcal{X}_t^+$. Observing that M is semantics preserving and by construction of M it follows that $\forall y \in \mathcal{X}_t^+$, $Pr_{y \in \mathcal{D}_t}[y] = Pr_{y \in \mathcal{D}_t^+}[y] = \sum_{x \in \mathcal{X}_s^+} \psi_{m_+}(x, y)$. Since we have already shown that when $y \in \mathcal{X}_t^+, Pr_{y \in m^x(\mathcal{D}_s)}[y] = \sum_{x \in \mathcal{X}_s^+} \psi_{m_+}(x, y)$ it follows that for $y \in \mathcal{X}_t^+, Pr_{y \in \mathcal{D}_t}[y] = Pr_{y \in m^x(\mathcal{D}_s)}[y]$. Proceeding in a similar way it can be shown that for $y \in \mathcal{X}_t^-, Pr_{y \in \mathcal{D}_t}[y] = Pr_{y \in m^x(\mathcal{D}_s)}[y]$. Hence the constructed mapping m^x and $\mathcal{D}_s$ induce $\mathcal{D}_t$. Consider the 1Delegating oracle $1EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X} \rangle$. As M is semantics preserving (by construction) it follows that $\phi = f$. Since we showed that m^x and $\mathcal{D}_s$ induce $\mathcal{D}_t$ it follows that 1Delegating oracle $1EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X} \rangle$ samples from $\mathcal{X}_t$ according to $\mathcal{D}_t$ and labels the sampled instance with f. Hence, $1EX\langle \phi, \{\mathcal{X}_s\}, \{\mathcal{D}_s\}, M, \mathcal{X} \rangle$ is indistinguishable from $EX(f, \mathcal{X}, \mathcal{D}_t)$.

APPENDIX C.

Often, in practice, there is a single available user mapping (say M_U assumed to be semantics preserving) that is used as a proxy for the true mapping M_{true}. In such a case it may be useful to get a bound on the error of using the mapping M_U as a proxy for the true mapping M_{true}. Since learning with a user provided mapping M_U instead of the true mapping M_{true} is the scenario $\mathcal{T} = \{kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle, EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle\}$, the error for which we are interested to provide a bound is $R(\mathcal{T})$ i.e. the error of the learning scenario $\mathcal{T}$. Since the learning scenario $\mathcal{T}$ is reducible to learning under domain shift by theorem (7), all the applicable positive bounds in domain adaptation (see Blitzer et al. (2007a); Ben-David et al. (2010)) are also applicable to $\mathcal{T}$. However, in our setting it is appropriate to provide a bound in terms of some notion of similarity between the mappings M_U and M_{true}.

Given $s(M_U)$ is a dataset obtained from a kDelegating oracle $kEX^m\langle\phi, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle$ using the mapping M_U, let $s(M_U \leftarrow M_{true})$ denote the dataset if the mapping M_{true} was used instead of M_U. Recall the notation that $S_\mathcal{X}$ is the set of all possible subset of $\mathcal{X}$ and $\mathcal{L}(s)$ is the hypothesis output by $\mathcal{L}$ when provided with dataset s.

Definition 14 ***μ-close Mapping**: Given a learning algorithm $\mathcal{L}$ and kDelegating Oracles $EX1\langle\phi_1, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle$ and $EX2\langle\phi_2, \mathcal{X}^+, \mathcal{D}^+, M_E, \mathcal{X}\rangle$, the mapping M_U is said to be μ-close to M_E iff $\forall s \in S_\mathcal{X},\;\; \epsilon_{M_U(\mathcal{D}^+)}(\mathcal{L}(s(M_E)), \mathcal{L}(s(M_E \leftarrow M_U))) \leq \frac{\mu}{2}$*

Observation. It follows directly from the definition that M_U being μ-close to M_E does not imply that M_E is μ-close to M_U.

Another quantity that we will use to bound $R(\mathcal{T})$ is the $\mathcal{A}$-*distance* that has been introduced and used in the setting of domain adaptation (Kifer et al., 2004; Blitzer et al., 2007a).

Given a hypothesis space $\mathcal{H}$, let $\mathcal{A}_{\mathcal{H}}$ be the set of subsets of $\mathcal{X}$ that support some hypothesis in $\mathcal{H}$. Hence, for every $h \in \mathcal{H}, \{x|x \in \mathcal{X}$ and $h(x) = 1\} \in \mathcal{A}_{\mathcal{H}}$. Given a set $A \subseteq \mathcal{X}$ let $Pr_{\mathcal{D}}[A]$ be the probability of obtaining set A when drawing randomly from $\mathcal{X}$ according to distribution $\mathcal{D}$ over $\mathcal{X}$

Definition 15 ***$\mathcal{A}$-distance (from Blitzer et al. (2007a))****: The A-distance between two distributions $\mathcal{D}_1$ and $\mathcal{D}_2$ over instance space $\mathcal{X}$ is defined as : $d_{\mathcal{H}}(\mathcal{D}_1, \mathcal{D}_2) = 2 \underset{A \in \mathcal{A}_{\mathcal{H}}}{sup} |Pr_{\mathcal{D}_1}[A] - Pr_{\mathcal{D}_2}[A]|$*

Definition 16 ***Symmetric Difference Hypothesis Space (from Blitzer et al. (2007a))****: Given a hypothesis space $\mathcal{H}$, the symmetric difference hypothesis space $\mathcal{H}\Delta\mathcal{H}$ as $\{h(x) \oplus h'(x)|h, h' \in \mathcal{H}\}$ where $\oplus$ is the XOR operator.*

It follows that given a hypothesis space $\mathcal{H}$, each hypothesis $g \in \mathcal{H}\Delta\mathcal{H}$ labels as positive all points x on which a given pair of hypotheses in $\mathcal{H}$ disagree. We can then define $\mathcal{A}_{\mathcal{H}\Delta\mathcal{H}}$ as the set of all sets A such that $A = \{x|x \in \mathcal{X}, h_1(x) \neq h_2(x)\}$ for some $h_1, h_2 \in \mathcal{H}$. We know state a useful inequality that has been first introduced in Blitzer et al. (2007a).

$$|\epsilon_{\mathcal{D}_1}(h, h') - \epsilon_{\mathcal{D}_2}(h, h')| \leq \frac{1}{2} d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_1, \mathcal{D}_2) \tag{C.1}$$

Theorem Given a learning algorithm $\mathcal{L}$ and a scenario $\mathcal{T} = \{kEX^m\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle, EX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle\}$ such that M_U is μ close to M_{true} and $f \in \mathcal{H}$ where $\mathcal{H}$ has VC-dimension v, then with probability $1 - \delta$

$$R(\mathcal{T}) \leq \hat{\epsilon}_{M_U(\mathcal{D}^+)}(\mathcal{L}(s(M_U)), f) + \mu + d_{\mathcal{H}\Delta\mathcal{H}}(M_U(\mathcal{D}^+), M_{true}(\mathcal{D}^+)) + \sqrt{\frac{v(log(2m/v)+1)+log(4/\delta)}{m}}$$

Proof 16 *Let h_U and h_T be the hypothesis output by $\mathcal{L}$ with access to oracles $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_U, \mathcal{X}\rangle$ and $kEX\langle f, \mathcal{X}^+, \mathcal{D}^+, M_{true}, \mathcal{X}\rangle$ respectively . To ease notation clutter let us represent the distribution $M_U(\mathcal{D}^+)$ by $\mathcal{D}_U$ and $M_{true}(\mathcal{D}^+)$ by $\mathcal{D}_T$. The proof relies on the use of triangle inequality for classification error: for a given distribution $\mathcal{D}$ and functions f_1, f_2 and f_3, it is the case that $\epsilon_{\mathcal{D}}(f_1, f_2) \leq \epsilon_{\mathcal{D}}(f_1, f_3) + \epsilon_{\mathcal{D}}(f_2, f_3)$ (see similar approach in Crammer et al. (2008); Blitzer et al. (2007a)). Hence,*

$$\begin{aligned}
\epsilon_{\mathcal{D}_T}(h_U, f) &\leq \epsilon_{\mathcal{D}_T}(h_U, h_T) + \epsilon_{\mathcal{D}_U}(h_T, f) \\
&\leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_U, \mathcal{D}_T) + \epsilon_{\mathcal{D}_U}(h_U, h_T) + \\
&\epsilon_{\mathcal{D}_U}(h_T, f) \textit{ (using C.1 twice)} \\
&\leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_U, \mathcal{D}_T) + 2\epsilon_{\mathcal{D}_U}(h_T, h_U) + \\
&\epsilon_{\mathcal{D}_U}(h_U, f) \textit{ (using triangle inequality)} \\
&\leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_U, \mathcal{D}_T) + \mu + \epsilon_{\mathcal{D}_U}(h_U, f) \\
&\textit{(using } M_{true} \textit{ and } M_U \textit{ are } \mu \textit{ close)} \\
&\leq d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_U, \mathcal{D}_T) + \hat{\epsilon}_{\mathcal{D}_U}(h_U, f) + \\
&\sqrt{\frac{(v(log(2m/v+1)) - log(\delta/4))}{m}} + \mu \\
&\textit{(using VC-dimension to relate} \\
&\textit{training error with true error)}
\end{aligned}$$

Since $R(\mathcal{T}) = \epsilon_{\mathcal{D}_T}(h_U, f)$*, the statement of the theorem follows.*

It is interesting to note that bound in Theorem depends on $\hat{\epsilon}_{\mathcal{D}_U}(h_U, f)$ (the training error), μ (that captures how close M_U is to M_{true}) and $d_{\mathcal{H}\Delta\mathcal{H}}(\mathcal{D}_U, \mathcal{D}_T)$ the symmetric hypothesis distance between the induced distributions by M_U and M_{true}.

Remark: We want to acknowledge that the bound in the stated theorem is inspired by similar domain adaptation bound in Blitzer et al. (2007a) (although our bound differs in that it incorporates the notion of closeness between mappings).

BIBLIOGRAPHY

(2009). Mappings of external classification systems to gene ontology. http://www.geneontology.org/GO.indices.shtml.

(2010). Ontology alignment evaluation initiative. Online. http://oaei.ontologymatching.org/.

Abouelhoda, M. I., Kurtz, S., and Ohlebusch, E. (2004). Replacing suffix trees with enhanced suffix arrays. *J. of Discrete Algorithms*, 2(1):53–86.

Agrawal, P., Sarma, A. D., Ullman, J., and Widom, J. (2010). Foundations of uncertain-data integration. In *VLDB 2010*.

Ai, W. I. and Langley, P. (1992). Induction of one-level decision trees. In *Proceedings of the Ninth International Conference on Machine Learning*, pages 233–240. Morgan Kaufmann.

Allende-Cid, H., Salas, R., Allende, H., and Ñanculef, R. (2007). Robust alternating adaboost. In *CIARP*, pages 427–436.

Alsabti, K., Ranka, S., and Singh, V. (1998). CLOUDS: A decision tree classifier for large datasets. In *Knowledge Discovery and Data Mining*, pages 2–8.

Angluin, D. and Laird, P. (1998). Learning from noisy examples. *Machine Learning*, 2:343–370.

Antoniou, G. and van Harmelen, F. (2008). *A Semantic Web Primer.* MIT Press, Cambridge, MA, 2. edition.

Aoun-Allah, M. and Mineau, G. (2007). Distributed data mining:why do more than aggregating models. In *IJCAI*.

Apweiler, R., Bairoch, A., and Wu, C. H. (2004). Protein sequence databases. *Current Opinion in Chemical Biology*, 8(1):76 – 80.

Armbrust, M., Fox, A., Griffith, R., Joseph, A. D., Katz, R., Konwinski, A., Lee, G., Patterson, D., Rabkin, A., Stoica, I., and Zaharia, M. (2010). A view of cloud computing. *Commun. ACM*, 53:50–58.

Bao, J., Caragea, D., and Honavar, V. (2007). Query translation for ontology extended data sources. AAAI Workshop on Semantic e-Science.

Bar-Or, A., Keren, D., Schuster, A., and Wolff, R. (2005). Hierarchical decision tree induction in distributed genomic databases. *IEEE Transactions on Knowledge and Data Engineering*, 17:1138–1151.

Begleiter, R., El-Yaniv, R., and Yona, G. (2004). On prediction using variable order markov models. *J. Artif. Intell. Res. (JAIR)*, 22:385–421.

Bejerano, G. and Yona, G. (2001). Variations on probabilistic suffix trees: statistical modeling and prediction of protein families. *Bioinformatics*, 17(1):23–43.

Ben-David, S., Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Vaughan, J. W. (2010). A theory of learning from different domains. *Machine Learning*, 79(1-2):151–175.

Ben-David, S., Blitzer, J., Crammer, K., and Pereira, F. (2007). Analysis of representations for domain adaptation. In *NIPS*.

Ben-david, S., Gehrke, J., and Schuller, R. (2002). A theoretical framework for learning from a pool of disparate data sources. In *In Proceedings of the The Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 443–449.

Berners-Lee, T., Hendler, J., and Lassila, O. (2001). The semantic web. Scientific American.

Beygelzimer, A., Dani, V., Hayes, T., Langford, J., and Zadrozny, B. (2005). Error limiting reductions between classification tasks. In *In Proceedings of the International Conference on Machine Learning*, pages 49–56.

Bhatnagar, R. and Srinivasan, S. (1997). Pattern discovery in distributed databases. In *AAAI/IAAI*, pages 503–508.

Bickel, S., Bruckner, M., and Scheffer, T. (2009). Discriminative learning under covariate shift. *Journal of Machine Learning Research*, 10:2137–2155.

Bishop, C. (2006). *Pattern Recognition and Machine Learning.*

Blitzer, J., Crammer, K., Kulesza, A., Pereira, F., and Wortman, J. (2007a). Learning bounds for domain adaptation. In *Advances in Neural Information Processing Systems*, volume 20.

Blitzer, J., Dredze, M., and Pereira, F. (2007b). Biographies, bollywood, boomboxes and blenders: Domain adaptation for sentiment classification. In *In ACL*, pages 187–205.

Bonatti, P., Deng, Y., and Subrahmanian, V. (2003). An ontology-extended relational algebra. In *IRI*, pages 192–199.

Borodovsky, M. and McIninch, J. D. (1993). Genmark: Parallel gene recognition for both dna strands. *Computers & Chemistry*, 17(2):123–133.

Breiman, L. (1984). *Classification and regression trees.* Chapman & Hall.

Breiman, L. (1999). Pasting small votes for classification in large databases and on-line. *Machine Learning*, 36:85–103.

Bshouty, N. H., Eiron, N., and Kushilevitz, E. (1999). Pac learning with nasty noise. *Theor. Comput. Sci.*, 2888:2002.

Burge, C. and Karlin, S. (1997). Prediction of complete gene structures in human genomic dna. *J. Mol. Biol*, 268:78–94.

Cafarella, M. J., Halevy, A., and Khoussainova, N. (2009). Data integration for the relational web. *Proc. VLDB Endow.*, 2:1090–1101.

Caragea, D. (2004). *Learning classifiers from distributed, semantically hetrogeneous, autonomous data sources.* PhD thesis, Iowa State University.

Caragea, D., Silvescu, A., and Honavar, V. . (2004a). A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems.*, 1:80–89.

Caragea, D., Silvescu, A., and Honavar, V. (2004b). A framework for learning from distributed data using sufficient statistics and its application to learning decision trees. *International Journal of Hybrid Intelligent Systems*, 1:2004.

Caragea, D., Zhang, J., Bao, J., Pathak, J., and Honavar, V. (2005). Algorithms and software for collaborative discovery from autonomous, semantically heterogeneous information sources (invited paper). In *Proceedings of the 16th International Conference on Algorithmic Learning Theory.*, volume 3734, pages 13–44. Springer-Verlag.

Chang, F., Dean, J., Ghemawat, S., Hsieh, W. C., Wallach, D. A., Burrows, M., Chandra, T., Fikes, A., and Gruber, R. E. (2006). Bigtable: a distributed storage system for structured data. In *OSDI '06: Proceedings of the 7th symposium on Operating systems design and implementation*, pages 205–218.

Chapelle, O., Schölkopf, B., and Zien, A., editors (2006). *Semi-Supervised Learning.* MIT Press, Cambridge, MA.

Chawathe, S., Garcia-Molina, H., Hammer, J., Ireland, K., Papakonstantinou, Y., Ullman, J. D., and Widom, J. (1994). The TSIMMIS Project: Integration of heterogeneous information sources. In *Proc. of the 16th Meeting of the Information Processing Society ofJapan.*

Chawla, N. V. and Karakoulas, G. (2005). Learning from labeled and unlabeled data: An empirical study across techniques and domains. *Journal of Artificial Intelligence Research*, 23:331–366.

Choi, N., Song, I.-Y., and Han, H. (2006). A survey on ontology mapping. *SIGMOD Rec.*, 35(3):34–41.

Chu, C. T., Kim, S. K., Lin, Y. A., Yu, Y., Bradski, G. R., Ng, A. Y., and Olukotun, K. (2006). Map-reduce for machine learning on multicore. In Schölkopf, B., Platt, J. C., and Hoffman, T., editors, *NIPS*, pages 281–288. MIT Press.

Clark, P. and Niblett, T. (1989). The cn2 induction algorithm. *Machine Learning*, 3:261–283.

Cortes, C., Mohri, M., Riley, M., and Rostamizadeh, A. (2008). Sample selection bias correction theory. In *ALT '08: Proceedings of the 19th international conference on Algorithmic Learning Theory*, pages 38–53.

Crammer, K., Kearns, M., and Wortman, J. (2008). Learning from multiple sources. *J. Mach. Learn. Res.*, 9:1757–1774.

Crawley, M. J. (2005). *Statistics: An Introduction using R.* Wiley.

Czarnowski, I. and Jedrzejowicz, P. (2008). Data reduction algorithm for machine learning and data mining. In *IEA/AIE '08: Proceedings of the 21st international conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems*, pages 276–285, Berlin, Heidelberg. Springer-Verlag.

Dai, W., Yang, Q., Xue, G. R., and Yu, Y. (2007). Boosting for transfer learning. In *ICML '07: Proceedings of the 24th international conference on Machine learning*, pages 193–200.

Daumé III, H. (2007). Frustratingly easy domain adaptation. In *Conference of the Association for Computational Linguistics (ACL)*, Prague, Czech Republic.

Daumé III, H. and Marcu, D. (2006). Domain adaptation for statistical classifiers. *Journal of Artificial Intelligence Research (JAIR)*, 26:101–126.

Davies, J., Fensel, D., and van Harmelen, F., editors (2002). *Towards the Semantic Web: Ontology-driven Knowledge Management.* Wiley.

DBLP (2010). D2r server publishing the dblp bibliography database. Online. http://www4.wiwiss.fu-berlin.de/dblp/.

Dean, J. and Ghemawat, S. (2008). Mapreduce: simplified data processing on large clusters. *Commun. ACM*, 51:107–113.

DeGroot, M. H. (2001). *Probability and Statistics.* Addison-Wesley.

Dietterich, T. (2000a). Ensemble methods in machine learning. In *Multiple Classifier Systems*, volume 1857 of *Lecture Notes in Computer Science*, pages 1–15. Springer Berlin / Heidelberg.

Dietterich, T. G. (2000b). An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Mach. Learn.*, 40(2):139–157.

Doan, A. and Halevy, A. Y. (2005). Semantic-integration research in the database community. *AI Mag.*, 26(1):83–94.

Doan, A., Madhavan, J., Domingos, P., and Halevy, A. (2002). Learning to map between ontologies on the semantic web. In *WWW '02: Proceedings of the 11th international conference on World Wide Web*, pages 662–673, New York, NY, USA. ACM.

Domingos, P. (1997). Knowledge acquisition from examples via multiple models. In *In Proceedings of the Fourteenth International Conference on Machine Learning*, pages 98–106. Morgan Kaufmann.

Dong, X., Halevy, A. Y., and Yu, C. (2007). Data integration with uncertainty. In *Proceedings of the 33rd international conference on Very large data bases*, pages 687–698.

Euzenat, J. and Shvaiko, P. (2007). *Ontology Matching.* Springer-Verlag.

Falconer, S., Noy, N., and Storey, M.-A. (2007). Ontology mapping-a user survey. In Shvaiko, P., Euzenat, J., Giunchiglia, F., and He, B., editors, *Proceedings of the Workshop on Ontology Matching (OM2007) at ISWC/ASWC2007,Busan,South Korea.*

Fayyad, U. M. and Irani, K. B. (1992). On the handling of continuous-valued attributes in decision tree generation. *Machine Learning*, 8:87–102.

Fischer, P., Larsen, S., and Thomsen, C. (2004). Predicting protein secondary structure with markov models. In *Proceedings of 29th Annual Conference of the German Classification Society*.

Flouris, G., Manakanatas, D., Kondylakis, H., Plexousakis, D., and Antoniou, G. (2008). Ontology change: Classification and survey. *Knowl. Eng. Rev.*, 23:117–152.

Foundation, A. S. (2010). Mahout 0.3. http://mahout.apache.org/.

Freund, Y. and Schapire, R. E. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119–139.

Gamberger, D., Lavrac, N., and Groselj, C. (1999). Experiments with noise filtering in a medical domain. In *ICML '99: Proceedings of the Sixteenth International Conference on Machine Learning*, pages 143–151, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Ghoting, A. and Pednault, E. (2009). Hadoop-ml: An infrastructure for the rapid implementation of parallel reusable analytics. NIPS workshop on Large-Scale Machine Learning: Parallelism and Massive Datasets.

Goh, C. H., Bressan, S., Madnick, S., and Siegel, M. (1999). Context interchange: new features and formalisms for the intelligent integration of information. *ACM Trans. Inf. Syst.*, 17(3):270–293.

Goldman, S. A. and Sloan, R. H. (1995). Can pac learning algorithms tolerate random attribute noise. *Algorithmica*, 14:70–84.

Grossman, R. (2001). Parallel methods for scaling data mining to large data sets. In *Handbook on Data Mining and Knowledge Discovery*. Oxford University Press.

Haas, L. (2007). The theory and practice of information integration. In *International Conference on Database Theory*.

Haas, L., Hentschel, M., Kossmann, D., and Miller, R. (2009). Schema and data: A holistic approach to mapping, resolution and fusion in information integration. In Laender, A., Castano, S., Dayal, U., Casati, F., and de Oliveira, J., editors, *Conceptual Modeling - ER 2009*, volume 5829 of *Lecture Notes in Computer Science*, pages 27–40. Springer Berlin / Heidelberg.

Halevy, A., Rajaraman, A., and Ordille, J. (2006). Data integration: the teenage years. In *Proceedings of the 32nd international conference on Very large data bases*, VLDB '06, pages 9–16.

Halevy, A. Y., Ashish, N., Bitton, D., Carey, M., Draper, D., Pollock, J., Rosenthal, A., and Sikka, V. (2005). Enterprise information integration: successes, challenges and controversies. In *Proceedings of the 2005 ACM SIGMOD international conference on Management of data*, pages 778–787.

Hall, L. O., Bowyer, K. W., Kegelmeyer, W. P., Moore, T. E., Jr., and ming Chao, C. (2000). Distributed learning on very large data sets. In *In Proceedings of the Sixth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 79–84.

Heckman, J. J. (1979). Sample selection bias as a specification error. *Econometrica*, 47(1):153–61.

Honavar, V. and Caragea, D. (2008). *Next Generation of Data Mining*, chapter Towards Semantics-Enabled Infrastructure for Knowledge Acquisition from Distributed Data. Taylor and Francis.

Hsu, K.-W., Banerjee, A., and Srivastava, J. (2008). I/o scalable bregman co-clustering. In *Proceedings of the 12th Pacific-Asia conference on Advances in knowledge discovery and data mining*, PAKDD'08, pages 896–903, Berlin, Heidelberg. Springer-Verlag.

Hull, R. (1997). Managing semantic heterogeneity in databases: a theoretical prospective. In *PODS '97: Proceedings of the sixteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems*, pages 51–61, New York, NY, USA. ACM.

Ingersoll, G. (2009). Introducing apache mahout. http://www.ibm.com/developerworks/java/library/j-mahout/index.html.

Jiang, J. (2008). *Domain Adaptation in Natural Language Processing*. PhD thesis, , University of Illinois at Urbana-Champaign.

Jiang, J. and Zhai, C. (2007). Instance weighting for domain adaptation in nlp. In *Proceedings of the 45th Annual Meeting of the Association of Computational Linguistics*, pages 264–271. Association for Computational Linguistics.

Jin, R. and Agrawal, G. (2003). Communication and memory efficient parallel decision tree construction. In *In Proceedings of Third SIAM Conference on Data Mining*.

John, G. H. (1995). Robust decision trees: Removing outliers from databases. In *In Knowledge Discovery and Data Mining*, pages 174–179. AAAI Press.

Kalfoglou, Y. and Schorlemmer, M. (2005). Ontology mapping: The state of the art. *Knowl. Eng. Rev*, 18(04391):1–31.

Kalyanpur, A., Parsia, B., Sirin, E., and Grau, B. C. (2006). Repairing unsatisfiable concepts in owl ontologies. In *ESWC*, pages 170–184.

Karmaker, A. and Kwek, S. (2005). A boosting approach to remove class label noise. *Hybrid Intelligent Systems, International Conference on*, 0:206–211.

Kearns, M. and Li, M. (1993). Learning in the presence of malicious errors. *SIAM Journal on Computing*, 22:267–280.

Kearns, M. J. and Vazirani, U. V. (1994). *An Introduction to Computational Learning Theory*. The MIT Press.

Kifer, D., Ben-David, S., and Gehrke, J. (2004). Detecting change in data streams. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 180–191.

Kotsiantis, S. B., Kanellopoulos, D., and Pintelas, P. E. (2006). Local boosting of decision stumps for regression and classification problems. *JCP*, 1(4):30–37.

Koul, N. (2008a). Indus integration framework. Google code - http://code.google.com/p/indusintegrationframework.

Koul, N. (2008b). Indus learning framework. Google Code - http://code.google.com/p/induslearningframework/.

Koul, N., Bui, N., and Honavar, V. (2010). Scalable, updatable predictive models for sequence data. In *BIBM*, pages 681–685.

Koul, N., Caragea, C., Honavar, V., Bahirwani, V., and Caragea, D. (2008). Learning classifiers from large databases using statistical queries. In *Web Intelligence*, pages 923–926.

Koul, N. and Honavar, V. (2009). Design and implementation of a query planner for data integration. In *ICTAI*, pages 214–218.

Koul, N. and Honavar, V. (2010). Learning in presence of ontology mapping errors. In *Web Intelligence*, pages 291–296.

Lenzerini, M. (2002). Data integration: a theoretical perspective. In *PODS '02: Proceedings of the twenty-first symposium on Principles of database systems*, pages 233–246, New York, NY, USA. ACM.

Liao, X., Xue, Y., and Carin, L. (2005). Logistic regression with an auxiliary data source. In *ICML '05: Proceedings of the 22nd international conference on Machine learning*, pages 505–512. ACM.

Liu, W. Z., White, A. P., Thompson, S. G., and Bramer, M. A. (1997). Techniques for dealing with missing values in classification. *Lecture Notes in Computer Science*, 1280:527–??

Manola, F. and Miller, E., editors (2004). *RDF Primer*. W3C Recommendation. World Wide Web Consortium.

Mansour, Y. (1997). Pessimistic decision tree pruning based on tree size. In *Proceedings of the Fourteenth International Conference on Machine Learning*, pages 195–201. Morgan Kaufmann.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2008). Domain adaptation with multiple sources. In *NIPS*.

Mansour, Y., Mohri, M., and Rostamizadeh, A. (2009). Domain adaptation: Learning bounds and algorithms. In *In Proceeding of COLT*.

Martin, J., Gibrat, J.-F., and Rodolphe, F. (2005). Choosing the optimal hidden markov model for secondary-structure prediction. *IEEE Intelligent Systems*, 20:19–25.

Martínez-Muñoz, G., Hernández-Lobato, D., and Suárez, A. (2007). Selection of decision stumps in bagging ensembles. In *ICANN (1)*, pages 319–328.

McCallum, A., Freitag, D., and Pereira, F. C. N. (2000). Maximum entropy markov models for information extraction and segmentation. In *ICML*, pages 591–598, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Meilicke, C., Stuckenschmidt, H., and Tamilin, A. (2007). Repairing ontology mappings. In *AAAI*, pages 1408–1413.

Meilicke, C., Stuckenschmidt, H., and Šváb Zamazal, O. (2009). A reasoning-based support tool for ontology mapping evaluation. In *ESWC 2009 Heraklion: Proceedings of the 6th European Semantic Web Conference on The Semantic Web*, pages 878–882, Berlin, Heidelberg. Springer-Verlag.

Mitchell, T. (1997). *Machine Learning*. McGraw-Hill, New York.

Molina, L., Belanche, L., and Nebot, A. (2002). Feature selection algorithms: a survey and experimental evaluation. In *ICDM*, pages 306–313.

Moore, A. and Lee, M. S. (1998). Cached sufficient statistics for efficient machine learning with large datasets. *Journal of Artificial Intelligence Research*, 8:67– 91.

Neville, J., Jensen, D., and Gallagher, B. (2003). Simple estimators for relational bayesian classifiers. In *ICDM*, pages 609–612.

Noy, N. F. (2009). Ontology mapping. In *Handbook on Ontologies*, International Handbooks on Information Systems, pages 573–590. Springer Berlin Heidelberg.

Pan, S. J. (2010). List of conferences and workshops where transfer learning papers appear.

Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on Knowledge and Data Engineering*, 22(10):1345–1359.

Park, B. and Kargupta, H. (2003). *The Handbook of Data Mining*, chapter Distributed Data Mining, pages 341–358. Lawrence Erlbaum Associates.

Park, B.-H. and Kargupta, H. (2002). *Distributed Data Mining*, chapter Distributed Data Mining: Algorithms, Systems and Applications, pages 341–358. LEA.

Polikar, R. (2006). Ensemble based systems in decision making. *IEEE Circuits and Systems Magazine*, 6(3):21–45.

Prodromidis, A. L., Chan, P. K., and Stolfo, S. J. (2000). Meta-learning in distributed data mining systems: Issues and approaches. In *Advances of Distributed Data Mining.* AAAI Press.

Provost, F. (2000). Distributed data mining: Scaling up and beyond.

Provost, F. and Kolluri, V. (1999). A survey of methods for scaling up inductive algorithms. *Data Min. Knowl. Discov.*, 3(2):131–169.

Quinlan, J. R. (1993). *C4.5: Programs for Machine Learning.* Morgan Kaufmann.

Rabiner, L. R. (1990). A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, 77:267–296.

Rokach, L. (2010). Ensemble-based classifiers. *Artif. Intell. Rev.*, 33:1–39.

Ron, D., Singer, Y., and Tishby, N. (1996). The power of amnesia. In *Machine Learning*, volume 6, pages 176–183.

Rothemund, P. W. K. (2001). *Theory and experiments in algorithmic self-assembly.* PhD thesis, Los Angeles, CA, USA. AAI3065842.

Salzberg, S., Delcher, A. L., Kasif, S., and White, O. (1998). Microbial gene identification using interpolated markov models. *Nucleic Acids Research*, 26:544–548.

Salzberg, S. L., Pertea, M., Delcher, A. L., Gardner, M. J., and Tettelin, H. (1999). Interpolated markov models for eukaryotic gene finding. *Genomics*, 59:24–31.

Shaban-Nejad, A. and Haarslev, V. (2009). Bio-medical ontologies maintenance and change management. In *Biomedical Data and Applications*, pages 143–168.

Shackelford, G. and Volper, D. (1988). Learning k-dnf with noise in the attributes. In *COLT '88: Proceedings of the first annual workshop on Computational learning theory*, pages 97–103, San Francisco, CA, USA. Morgan Kaufmann Publishers Inc.

Shafer, J. C., Agrawal, R., and MehtaShafer1996, M. (1996). SPRINT: A scalable parallel classifier for data mining. In Vijayaraman, T. M., Buchmann, A. P., Mohan, C., and Sarda, N. L., editors, *Proc. 22nd Int. Conf. Very Large Databases, VLDB*, pages 544–555. Morgan Kaufmann.

Shvaiko, P. and Euzenat, J. (2005). A survey of schema-based matching approaches. *Journal on Data Semantics*, 4:146–171.

Shvaiko, P. and Euzenat, J. (2008). Ten challenges for ontology matching. In Meersman, R. and Tari, Z., editors, *On the Move to Meaningful Internet Systems*, volume 5332, chapter 18, pages 1164–1182. Springer Berlin Heidelberg.

Singh, A., Nowak, R. D., and Zhu, X. (2008). Unlabeled data: Now it helps, now it doesn't. In *NIPS*, pages 1513–1520.

Sloan, R. H. (1995). Four types of noise in data for pac learning. *Inf. Process. Lett.*, 54(3):157–162.

Stanke, M. and Waack, S. (2003). Gene prediction with a hidden markov model and a new intron submodel. *Bioinformatics*, 19:215–225.

Stein, L. (2010). The case for cloud computing in genome informatics. *Genome Biology*, 11(5):207+.

Stonebraker, M. (2005). One size fits all: An idea whose time has come and gone. In *In Proceedings of the International Conference on Data Engineering (ICDE*, pages 2–11.

Sun, Z. and Deogun, J. S. (2004). Local prediction approach for protein classification using probabilistic suffix trees. In *APBC '04: Proceedings of the second conference on Asia-Pacific bioinformatics*, pages 357–362.

Tauberer, J. (2010). The 2000 u.s. census: 1 billion rdf triples. Online. http://www.rdfabout.com/demo/census/.

Tveit, A. and Engum, H. (2003). Parallelization of the incremental proximal support vector machine classifier using a heap-based tree topology. Parallel and Distributed Computing For Machine Learning.

Valiant, L. G. (1984). A theory of the learnable. *Commun. ACM*, 27(11):1134–1142.

Verbaeten, S. and Assche, A. V. (2003). Ensemble methods for noise elimination in classification problems. In *Multiple Classifier Systems*, pages 317–325.

Visser, U., Stuckenschmidt, H., Wache, H., and Vgele, T. (2000). Enabling technologies for interoperability. In *TZI, University of Bremen*, pages 35–46. TZI.

Vouk, M. (2008). Cloud computing - issues, research and implementations. In *Information Technology Interfaces, 2008. ITI 2008. 30th International Conference on.*

W3C (2010a). R2rml: Rdb to rdf mapping language. Online. http://www.w3.org/TR/2010/WD-r2rml-20101028/.

W3C (2010b). Sparql 1.1 query language. Online. http://www.w3.org/TR/sparql11-query/.

Wache, H. and Stuckenschmidt, H. (2001). Practical context transformation for information system interoperability. In *CONTEXT '01: Proceedings of the Third International and Interdisciplinary Conference on Modeling and Using Context*, pages 367–380, London, UK. Springer-Verlag.

Wache, H., Vögele, T., Visser, U., Stuckenschmidt, H., Schuster, G., Neumann, H., and Hübner, S. (2001). Ontology-based integration of information — a survey of existing approaches. In Stuckenschmidt, H., editor, *IJCAI–01 Workshop: Ontologies and Information Sharing*, pages 108–117.

Wang, L., Tao, J., Kunze, M., Castellanos, A., Kramer, D., and Karl, W. (2008). Scientific cloud computing: Early definition and experience. In *High Performance Computing and Communications, 2008. HPCC '08. 10th IEEE International Conference on*, pages 825 – 830.

Wang, L., von Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., and Fu, C. (2010). Cloud computing: a perspective study. *New Generation Computing*, 28:137–146.

Wilson, D. R. and Martinez, T. R. (2000). Reduction techniques for instance-basedlearning algorithms. *Mach. Learn.*, 38(3):257–286.

Wilson, D. R. and Martinez, T. R. (2004). Reduction techniques for exemplar-based learning algorithms. In *Machine learning*, pages 257–286.

Witten, I. H. and Frank, E. (2005). *Data Mining: Practical machine learning tools and techniques.* Morgan Kaufmann.

Wu, P. and Dietterich, T. G. (2004). Improving svm accuracy by training on auxiliary data sources. In *ICML '04: Proceedings of the twenty-first international conference on Machine learning.*

Yakhnenko, O., Silvescu, A., and Honavar, V. (2005). Discriminatively trained markov model for sequence classification. In *ICDM '05: Proceedings of the Fifth IEEE International Conference on Data Mining*, pages 498–505.

Yang, Q. (2009). Transfer learning beyond text classification. In Zhou, Z.-H. and Washio, T., editors, *Advances in Machine Learning*, volume 5828 of *Lecture Notes in Computer Science*, pages 10–22. Springer Berlin / Heidelberg.

Yuan, Z. (1999). Prediction of protein subcellular locations using markov chain models. *FEBS Letters*, pages 23–26.

Zadrozny, B. Z. (2004). Learning and evaluating classifiers under sample selection bias. In *In International Conference on Machine Learning ICML04*, pages 903–910.

Zaki, M. J. (1999). Parallel and distributed association mining: A survey. *IEEE Concurrency*, 7(4):14.

Zhu, H., Wang, J., Yang, Z., and Song, Y. (2006a). Interpolated hidden markov models estimated using conditional ml for eukaryotic gene annotation. In *Computational Intelligence and Bioinformatics*, pages 267–274. Springer Berlin / Heidelberg.

Zhu, X. and Wu, X. (2004). Class noise vs. attribute noise: A quantitative study. *Artif. Intell. Rev.*, 22(3):177–210.

Zhu, X., Wu, X., and Chen, Q. (2003). Eliminating class noise in large datasets. In *ICML*, pages 920–927.

Zhu, X., Xingqua, N., Wu, Xindon, G., Che, N., and Qiju, N. (2006b). Bridging local and global data cleansing: Identifying class noise in large, distributed data datasets. *Data Mining and Knowledge Discovery*, 12(2-3):275–308.

Ziegler, P. and Dittrich, K. R. (2004). User-specific semantic integration of heterogeneous data: The sirup approach. In *First International IFIP Conference on Semantics of a Networked World (ICSNW 2004)*.

Ziegler, P. and Dittrich, K. R. (2007). Data integration — problems, approaches, and perspectives. In Krogstie, J., Opdahl, A. L., and Brinkkemper, S., editors, *Conceptual Modelling in Information Systems Engineering*, pages 39–58. Springer, Berlin.

Zou, B., Ma, X., Kemme, B., Newton, G., , and Precup, D. (2006). Data mining using relational database management systems. In *Advances in Knowledge Discovery and Data Mining*, volume 3918 of *Lecture Notes in Computer Science*, pages 657–667. Springer Berlin / Heidelberg.